COMPUTER-AIDED DESIGN OF PASSIVE FILTERS IN THE FREQUENCY DOMAIN USING GRADIENT-PROJECTION MINIMIZATION TECHNIQUES

by

Charles Alden Henry

United States
Naval Postgraduate School

THESIS

COMPUTER-AIDED DESIGN
OF PASSIVE FILTERS IN THE FREQUENCY DOMAIN
USING GRADIENT-PROJECTION MINIMIZATION TECHNIQUES

by

Charles Alden Henry

December 1969

Computer-Aided Design of Passive Filters in the Frequency Domain Using Gradient-Projection Minimization Techniques

by

Charles Alden Henry
Major, United States Marine Corps
B.S., United States Naval Academy, 1955

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1969

ABSTRACT

An algorithm for the computer-aided design of passive electrical filters using minimization techniques is presented. The configuration, or topology of the filter and its desired frequency response are specified. A measure of the deviation between the desired and the actual frequency response is minimized using the gradient projection search method. Examples are presented which demonstrate the performance of the minimization procedure.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

ACKNOWLEDGEMENT

The author is indebted to Professor D. E. Kirk for originally suggesting the problem and for providing guidance, assistance, and encouragement throughout the investigation.

I. INTRODUCTION

The development of extremely high-speed computers and efficient algorithms has opened the door to the use of iteration techniques that heretofore have been prohibitively costly in time and/or money. This high-speed capability makes it possible to consider a computer-aided solution to the problem of designing an electrical filter to match — as closely as possible — a given frequency response, wherein many hundreds — sometimes thousands — of successive solutions of the network frequency response are required. Use of the designing engineer's experience in adaptation of filters whose response is already known, but is not quite that desired, usually demands laborious hand calculation. In addition, there is occasional difficulty in achieving the required "closeness of fit" to the desired response. Some frequency response requirements may lie beyond the capability of synthesis techniques, being solvable only by experience coupled with trial and error procedures — which may never produce the desired accuracy.

With the availability of a computer program capable of systematically minimizing deviations from the required response, design time can be cut dramatically. The ease with which this tool can produce results has multiple benefits:

1. The engineer is free to concentrate on design without the tedium of repetitious computation.
2. The designer can investigate numerous circuit configurations to determine one which "best" fits the needs of simplicity, cost, and quality of performance.
3. The computer's speed enables the designer to quickly obtain an answer.

II. OPTIMIZATION THEORY

A. MINIMIZING THROUGH OPTIMIZATION

Optimization techniques are used to provide the means of systematically and effectively developing the solution to the problem. Classical methods for designing filters to meet a given frequency response are highly time-consuming, and usually produce only approximations of the desired response, perhaps matching the general shape, and a specified bandwidth [Ref. 1]. Optimization techniques utilize the designer's particular definition for "closeness of fit" of the designed to the specified frequency response and then proceed through successive adjustments of the variable parameters to minimize the deviation. The function defined by the designer to specify this closeness of fit will be referred to as "the cost function".

A specific filter configuration is required by the optimization process; no provision is made to permit computer-determined alteration of the circuit in an effort to improve the overall closeness of fit. This decision is left to the engineer to be carried out during his interaction with the program.

B. MINIMIZATION TECHNIQUES

Three methods for minimization were considered: direct, or pattern search; the gradient method; and second-order methods.

1. Pattern Search

The pattern-search method makes use of successive perturbations of the variable parameters to determine when a lower value of the cost function is achieved. When a reduction in value of the cost function is determined, the new point is then used to again perturb the parameters,

taking larger steps when continued successes are achieved. If a failure to decrease the cost function occurs, the step size is reduced until success is achieved. This is the simplest of the three methods, requiring only values of the cost function, but is somewhat sensitive to starting point when local minima are present [Ref. 2].

2. Gradient Method

The gradient method makes use of the negative of the first partial derivative of the cost function with respect to each of the variable parameters; this determines the direction of steepest descent. A step of specified distance is taken in the steepest-descent direction to decrease the value of the cost function.

A closely related method, that of gradient projection [Ref. 3], makes use of the negative gradient at the starting point to extend a vector in this "steepest-descent" direction until one of the parameter constraints is reached. The smallest cost function value along this extended vector is then determined. The negative gradient vector is computed for this new point, and the previous step is repeated until the minimum is reached. The algorithm will always move in the negative gradient direction unless doing so causes a violation of the constraints, in which case the negative gradient vector is projected onto the constraint boundary, and the next step is taken in the direction of this projection.

3. Second-Order Methods

Second-order methods involve the use of both the first and second partial derivatives of the cost function, and were not considered once the difficulty in generating the second partial derivative for filter networks was recognized.

C. PROBLEM CHARACTERISTICS

1. The Cost Function

The general format for the cost function, $J(\underset{\sim}{p})$, used in the optimization scheme is defined as:

$$J(\underset{\sim}{p}) = \int_{\omega_{min}}^{\omega_{max}} h(\omega) \, |e(\underset{\sim}{p})|^N \, d\omega \qquad (2.1)$$

Here $\underset{\sim}{p}$ is the vector of variable circuit parameters; $h(\omega)$ is a weighting function used to force the solution to more closely fit the prescribed response in certain critical areas; $e(\underset{\sim}{p})$ is the error between the desired and the actual frequency response; N represents the power to which e(p) is raised to determine the type of closeness of fit (e.g., N = 2 for mean-square), and normally takes on the value of 1, 2 or 4; ω_{min} and ω_{max} are the lower and upper frequencies over which the integration is performed.

The integration must be written in discrete form for use in a computer program; thus the following approximation is used:

$$J(\underset{\sim}{p}) = \Delta\omega \sum_{k=1}^{K} h(\omega_{min}+k\Delta\omega)|R_d(\omega_{min}+k\Delta\omega)-R_a(\omega_{min}+k\Delta\omega)|^N . \qquad (2.2)$$

In this equation $\Delta\omega$ is the difference between successive frequency points; K represents the number of frequency points to be matched; N is as described above, and $h(\omega_{min}+k\Delta\omega)$ is the discrete equivalent of $h(\omega)$; $R_d(\omega_{min}+k\Delta\omega)$ is the desired frequency response at the k^{th} frequency point, while $R_a(\omega_{min}+k\Delta\omega)$ is the actual response. It is this function, $J(\underset{\sim}{p})$, that the optimization process is to minimize. If a perfect fit to the desired frequency response is achieved, the value of $J(\underset{\sim}{p})$ will be zero, its absolute minimum.

2. Termination Criteria

Iteration of an optimization program can be terminated in several ways. Three of these are:

1. When further iteration produces no meaningful improvement in closeness of fit to the desired frequency response, iteration is terminated. A suitable equation is:

$$J^{(i)}(\underset{\sim}{p}) - J^{(k+1)}(\underset{\sim}{p}) < \varepsilon \qquad (2.3)$$

where ε is some relatively small positive number. If ε is selected to be too small, excessive iterations will usually be required to produce two successive values of $J(\underset{\sim}{p})$ close enough together to terminate the procedure, resulting in wasted computer time. If a large value is chosen for ε, the program may terminate far from a minimum.

2. When the norm of the gradient vector is quite close to zero, either a minimum, a maximum or a saddle point has been found; in a constrained problem, maximums are less likely. Numerous problems were run, and neither a maximum nor a saddle point ever resulted as an optimization solution. This criterion can be written:

$$\left\| \frac{\partial J(\underset{\sim}{p})}{\partial \underset{\sim}{p}} \right\| < \gamma$$

where γ is a suitably small positive number.

3. A third criterion, not written into the computer program used in this investigation, but one which might prove useful to the designer in utilization of the program is:

$$J^{(i)}(\underset{\sim}{p}) < \delta \qquad (2.5)$$

where δ is some suitably small positive number. Inclusion of this inequality into the computer program would cause termination once the cost function reaches a value less than δ.

3. Constraints

In many practical design problems, there is some idea of a reasonable range of constraints for each variable parameter, therefore it is assumed that constraints of the form shown below can be imposed on the n variable parameters:

$$a_i \leq p_i \leq b_i \qquad i = 1,2,\ldots,n \tag{2.6}$$

where $\underset{\sim}{a} = (a_1,a_2,\ldots,a_n)$ represents the vector lower limit, and $\underset{\sim}{b} = (b_1,b_2,\ldots,b_n)$ is the vector upper limit. In the case of several variables, the constraints influence the rate of convergence, especially for the gradient-projection method. For example, a reduction of constraints from $0. \leq p_n \leq 1.$ to $0. \leq p_n \leq .5$, where n = 5, will reduce a five-dimensional space by 96.9 percent (using $(1/2)^5 = .03124$), leaving a smaller region in which to search for the minimum.

III. THE OPTIMIZATION PROGRAM

A. THE FREQUENCY ANALYSIS PROGRAM

Several existing computer programs for calculating the frequency response of a given circuit were studied. Some of these were:

Calahan
Sceptre
Ecap
Cornap

The Calahan Network Analysis Program was chosen as being best able to provide the required data, in the desired form, with high accuracy and reasonable speed.

1. The Basic Program

The Linear Network Analysis Program by D. A. Calahan [Ref. 4] is a large general purpose program capable of producing a variety of circuit information. Inputs to the program are: either the network transfer function or a topological description; keys to indicate the desired output; high- and low-frequency limits and the number of points to be determined, if the frequency response is desired; initial and final values, initial and final time, and the time increment if a time response is desired. Provisions are made for varying one element for as many as eight different values. The output from the program includes the coefficients of the network function, the poles and zeros, the frequency response, the time response (to an arbitrary input), and the symbolic form of the network function. The program is capable of accepting circuits with as many as thirty nodes, one hundred passive elements, and twenty active elements. Printer-plot graphs of frequency and time response are produced when

desired. Negative element values, as well as negative frequencies, are acceptable to the program.

2. Accuracy

Examination of the accuracy of the frequency response produced by the Calahan analysis was accomplished in the following fashion. Specific circuits whose frequency response could be readily calculated using formulae (Butterworth and Chebychev low-pass filters in particular were used, producing the expected frequency response using formulae found in [Ref. 5]) were tested using the nominal element values. The Calahan results compared favorably with the answers from the formulae. For example, analysis of a five-element Butterworth filter gave the following results:

Freq(Hz)	Formula Gain(db)	Calahan Gain(db)
.15	-2.10196	-2.10181
.16	-3.10327	-3.10314
.17	-4.30471	-4.30461
.18	-5.65472	-5.65465
.19	-7.09729	-7.09723
.20	-8.58441	-8.58439

3. Modifications

Because only the frequency response was desired as an output from Calahan, and this on a highly repetitive basis, modifications to the program were made. Several subroutines and the main program were removed; a new main program was substituted so that the solution to the minimization problem could be generated by repeated use of the Calahan subroutines to determine the required frequency response. Two of the remaining Calahan subroutines were modified to conform to the desired iteration scheme.

B. THE MINIMIZATION PROGRAM

The gradient, or steepest-descent method, was chosen to be used for

minimization of the previously specified cost function. (The pattern search solution to this problem is the subject of a Naval Postgraduate School thesis by Captain James Lau, USMC. Comparative results are discussed in Chapter VI.)

1. Determination of the Gradient Vector

Some thought was given to the best method of producing the vector $\frac{\partial J(\underset{\sim}{p})}{\partial \underset{\sim}{p}}$. The literal matrix of partial derivatives was manually computed for three- and five-element filters, and it was quickly seen that this method is much too cumbersome and time-consuming, and will greatly reduce the ease with which the designer can use the program. As an example, given below is one partial for a three-element filter (two variable elements, L and C; R=1).

$$\frac{\partial J(L,C)}{\partial L} = \sum_{k=1}^{K} \frac{-2[LC^2(k\Delta\omega)^4+L(k\Delta\omega)^2-c(k\Delta\omega)^2]}{[L^2C^2(k\Delta\omega)^4+(L^2-2LC)(k\Delta\omega)^2+1]^2}$$

$$\left\{\frac{1}{L^2C^2(k\Delta\omega)^4+(L^2-2LC)(k\Delta\omega)^2+1} - \frac{1}{1+(k\Delta\omega)^4}\right\} . \qquad (3.1)$$

It was then decided to investigate the method of finite-difference approximation. This method makes use of the definition of the partial derivative,

$$\frac{\partial J(\underset{\sim}{p})}{\partial p_1} = \lim_{\Delta p_1 \to 0} \frac{J(p_1+\Delta p_1, p_2, \ldots, p_n) - J(p_1, p_2, \ldots, p_n)}{\Delta p_1} . \qquad (3.2)$$

The finite-difference approximation assigns a small but finite value to Δp_1 and determines the approximate partial derivative as:

$$\frac{\partial J(\underset{\sim}{p})}{\partial p_1} = \frac{J(p_1+\Delta p_1, p_2, \ldots, p_n) - J(p_1, p_2, \ldots, p_n)}{\Delta p_1} . \qquad (3.3)$$

Experimentation with various networks, using the Calahan subroutines to produce the frequency response indicated that a Δp of 0.01 percent gives reasonably accurate results. This accuracy was determined in two ways: by comparison with the values of the partial derivative vector for a two-variable-element filter produced by literal equations, one of which is equation (3.1); by use of discrete approximation to produce the gradient vector at the known absolute minimum of various larger networks. The first method produced normalized discrete approximation vectors within 1.5 percent of the true partial derivative vector in each of the ten points compared. The second method utilized larger networks whose actual frequency response was known, as well as the element values necessary to produce this response. When the correct element values were entered into the optimization program, each element of the gradient vector was less than 5×10^{-4}, in all examples tested.

2. The Gradient Method

Initially, a number of circuit optimization problems were tried, using the gradient method. In this program, new values for the variables are produced using the following iteration equation:

$$\underset{\sim}{p}^{(i+1)} = \underset{\sim}{p}^{(i)} - \beta \frac{\dfrac{\partial J(\underset{\sim}{p}^{(i)})}{\partial \underset{\sim}{p}^{(i)}}}{\left\| \dfrac{\partial J(\underset{\sim}{p}^{(i)})}{\partial \underset{\sim}{p}^{(i)}} \right\|} \quad . \tag{3.4}$$

Each iteration produces a new set of parameter values, where the normalized gradient vector is "stepped" a distance β. When the new value for the cost function is greater than the previous value, β is halved and the step is repeated until a smaller function value is obtained. This method

produces good results on filters with up to five variable parameters, but required many iterations that reduce the value of the cost function very slowly as the minimum is approached. A better method was therefore sought.

2. The Gradient-Projection Method

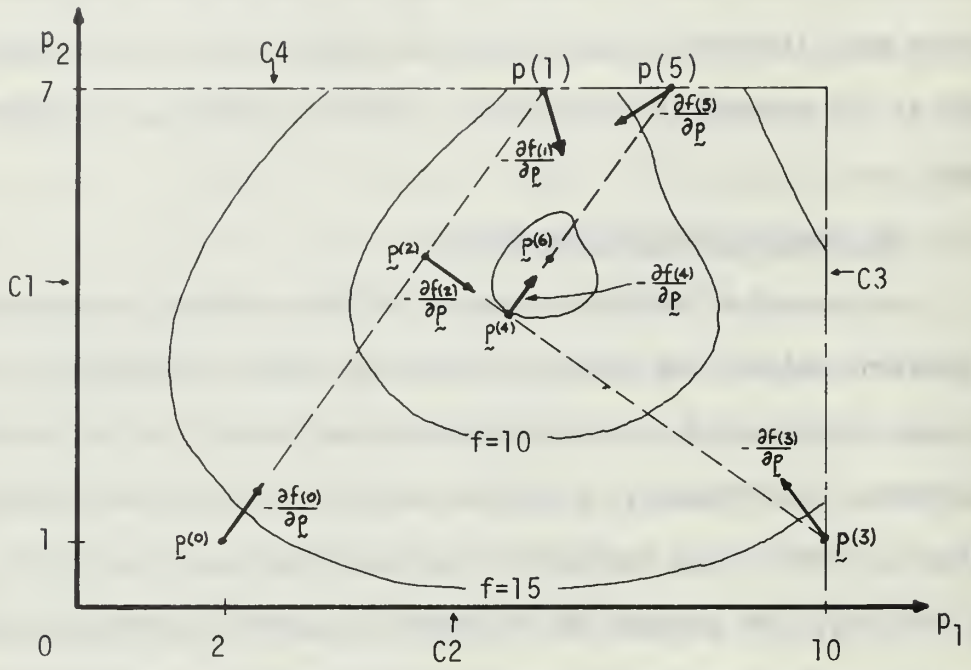
In hope of eliminating some of the disadvantages encountered in the gradient method, the gradient-projection method introduced by J. B. Rosen was investigated. This algorithm showed promise of more rapid convergence. Additionally, a program using this method was locally available. This is the Gradient Projection Program written by Dr. D. E. Kirk [Ref. 6]. The process is an iterative numerical method for locating the extremum of a function of many variables which are required to satisfy linear constraints. Unconstrained variables cannot be used with this method, as will be shown.

A simple two-dimensional example will serve to illustrate the use of this method. Let $f(\underset{\sim}{p})$ be a differentiable function of two variables p_1 and p_2; $f(p_1,p_2)$ represents the value of the function at the point (p_1,p_2). The objective is to find the point $(P_1{*},p_2{*})$ where $f(\underset{\sim}{p}^*)$ is a minimum, and which satisfies the constraints:

$$0. \leq p_1 \leq 10.$$

$$0. \leq p_2 \leq 7.$$

Figure 1(a) represents a possible contour plot of $f(\underset{\sim}{p})$ as a function of p_1 and p_2; the region which satisfies the constraints lies within the rectangle bounded by the lines C1, C2, C3 and C4. The starting point is at $\underset{\sim}{p}(0) = (2,1)$. At this point, the negative gradient vector, $-\frac{\partial f(0)}{\partial p}$, is calculated, and extended until a constraint boundary is reached, in



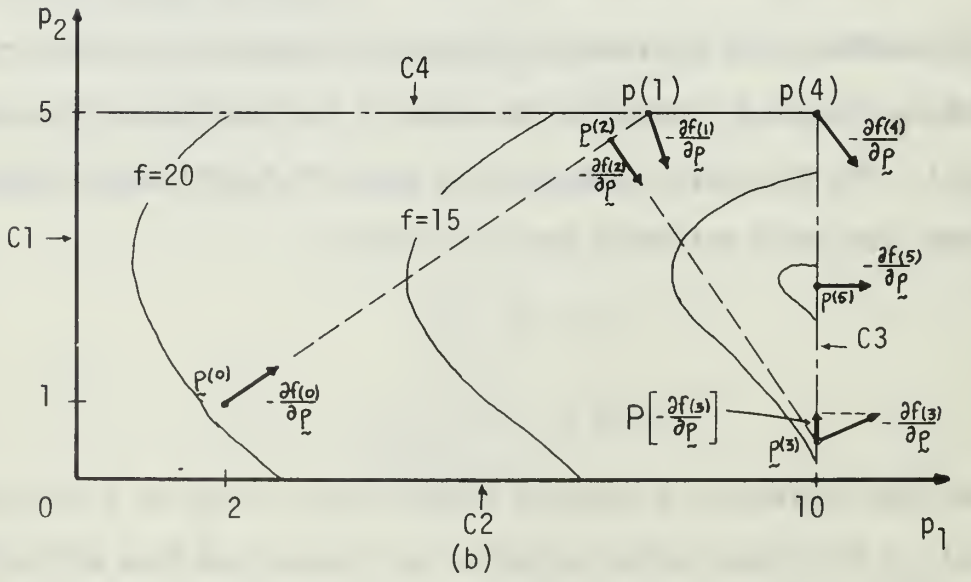


Figure 1

Contour Maps Demonstrating the Gradient Projection Method

this case C4. The gradient at $\underset{\sim}{p}(1)$, the intersection of C4 and the extended vector $-\frac{\partial f(0)}{\partial \underset{\sim}{p}}$, is calculated. The inner product of $-\frac{\partial f(0)}{\partial \underset{\sim}{p}}$ and $-\frac{\partial f(1)}{\partial \underset{\sim}{p}}$ is negative, indicating that the minimum value of $f(\underset{\sim}{p})$ along this extended vector is somewhere between $\underset{\sim}{p}(0)$ and $\underset{\sim}{p}(1)$; therefore interpolation is necessary. Successive interpolation locates the point $\underset{\sim}{p}(2)$, where $-\frac{\partial f(2)}{\partial \underset{\sim}{p}}$ is orthogonal to the originally extended vector. Orthogonality is indicated when the inner product of $-\frac{\partial f(0)}{\partial \underset{\sim}{p}}$ and $-\frac{\partial f(2)}{\partial \underset{\sim}{p}}$ is zero. The negative gradient vector $-\frac{\partial f(2)}{\partial \underset{\sim}{p}}$ is then extended to the nearest constraint boundary, and the process continues until the minimum is reached at $\underset{\sim}{p}(6)$, as shown in Figure 1(a).

Figure 1(b) shows a different hypothetical contour plot for $f(\underset{\sim}{p})$, using the constraints:

$$0. \leq p_1 \leq 10.$$

$$0. \leq p_2 \leq 5.$$

In this case, the absolute minimum lies outside the allowable constrained region; the minimum therefore lies along a boundary. This situation causes the constraint boundary C3 to be "active"; that is, it separates the acceptable region (that region within the constraint boundaries) from the absolute minimum of the function $f(\underset{\sim}{p})$. Once again, the starting point is $\underset{\sim}{p}(0) = (2,1)$. As before, the inner product of $-\frac{\partial f(0)}{\partial \underset{\sim}{p}}$ and $-\frac{\partial f(1)}{\partial \underset{\sim}{p}}$ is negative, indicating that interpolation is necessary. The point $\underset{\sim}{p}(2)$ is found and its negative gradient extended to the boundary C3 where, unlike the preceeding step, the inner product of $-\frac{\partial f(2)}{\partial \underset{\sim}{p}}$ and $-\frac{\partial f(3)}{\partial \underset{\sim}{p}}$ is positive, indicating that $\underset{\sim}{p}(3)$ is the minimum along the extension of $-\frac{\partial f(1)}{\partial \underset{\sim}{p}}$ and that no interpolation is necessary. However, a change of $\underset{\sim}{p}$ in the direction of $-\frac{\partial f(3)}{\partial \underset{\sim}{p}}$ would cause a violation of the constraints; therefore $\underset{\sim}{p}$ is changed along the vector represented by the projection of

$-\frac{\partial f(3)}{\partial \tilde{p}}$ along the boundary C3. This projection is denoted by $P[-\frac{\partial f(3)}{\partial \tilde{p}}]$, and is extended until another constraint is intersected, this time C4. Here the inner product of $P[-\frac{\partial f(3)}{\partial \tilde{p}}]$ and $-\frac{\partial f(4)}{\partial \tilde{p}}$ is negative, requiring interpolation. Successive interpolations locate $\tilde{p}(5)$, where $P[-\frac{\partial f(3)}{\partial \tilde{p}}]$ and $-\frac{\partial f(5)}{\partial \tilde{p}}$ are orthogonal. This is the minimum within the acceptable region. It should now be evident that unconstrained parameters cannot be used with this method.

In Figures 1(a) and (b), the functions to be minimized are unimodal. If, however, a given function is not unimodal within the acceptable region, then more than one minimum can exist, and the minimum found by the gradient-projection method may depend on the starting point. Figure 2 demonstrates this, showing solutions for two different starting points (gradient vectors are not labeled in order to enhance clarity). A point of interest is demonstrated by the projection vector from the second starting point: the actual minimum along this projection vector is located at A; however, the inner-product method indicates that no interpolation is necessary, and accepts point $\tilde{p}'(1)$ as the next starting point. This clearly demonstrates the requirement of unimodality for a function to be minimized by this interpolation method if the absolute minimum is desired, and multiple starting points are to be avoided.

A more detailed explanation of this method, including the matrix equations required to be solved in the computer program, is contained in Reference 6.

4. Cost-Function Complexity

The inner-product interpolation scheme incorporated in the gradient projection program produces accurate results when a convex function is to be minimized; a convex function is unimodal. In using this interpolation method to optimize the frequency response of a filter network, it was found

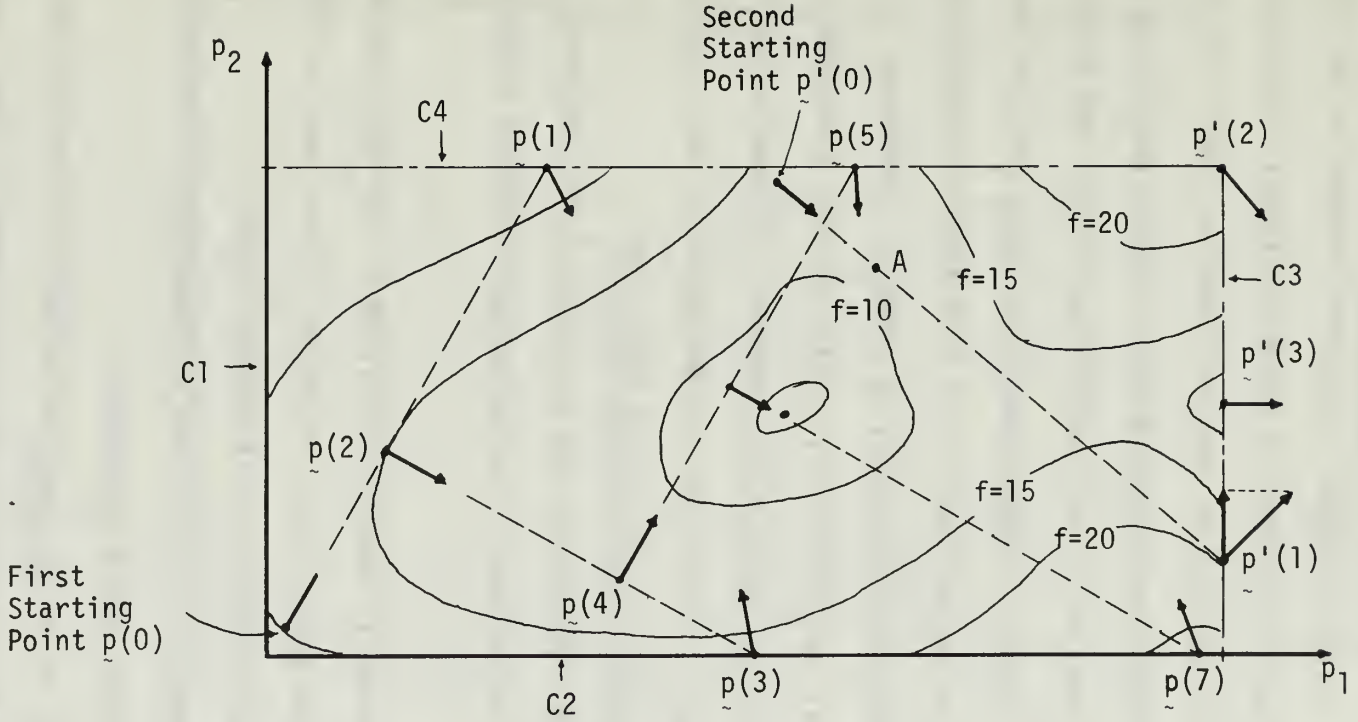


Figure 2

Contour Map Showing Location of Two Different Minima by the Gradient Projection Method

that the scheme was not functioning so that ever-decreasing values of the cost function were obtained. This helped confirm what was already suspected: the cost function, $J(\underset{\sim}{p})$, as applied to filter network frequency response may not be a convex function. An actual contour map of $J(\underset{\sim}{p})$ as a function of two variable parameters is shown in Figures 3(a) and (b). This map was produced by solving for the cost function over a grid of values of the two variables and joining points of equal cost-function value. The map shows that while only one minimum exists, the function $J(\underset{\sim}{p})$ is obviously not convex. Higher-dimensional maps cannot be easily drawn, but experience in minimizing filters with several variable elements indicates that more than one minimum may be present. Therefore it is concluded that the cost function, $J(\underset{\sim}{p})$, as used in this procedure is not convex, and is possibly, in general, not unimodal. No conclusive results in this area were sought, as this was not the object of the investigation.

5. The Function-Search Interpolation Procedure

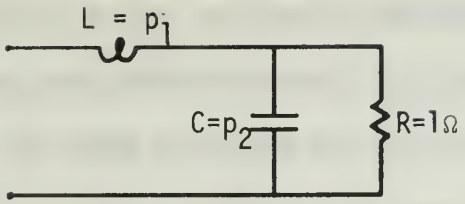
Armed with the knowledge of the cost function's complexity, a new interpolation method was sought. This method would ideally achieve the following goals:

Goal 1. Continually decrease the value of the cost function, never accepting any new point unless it yields a lower value of $J(\underset{\sim}{p})$.

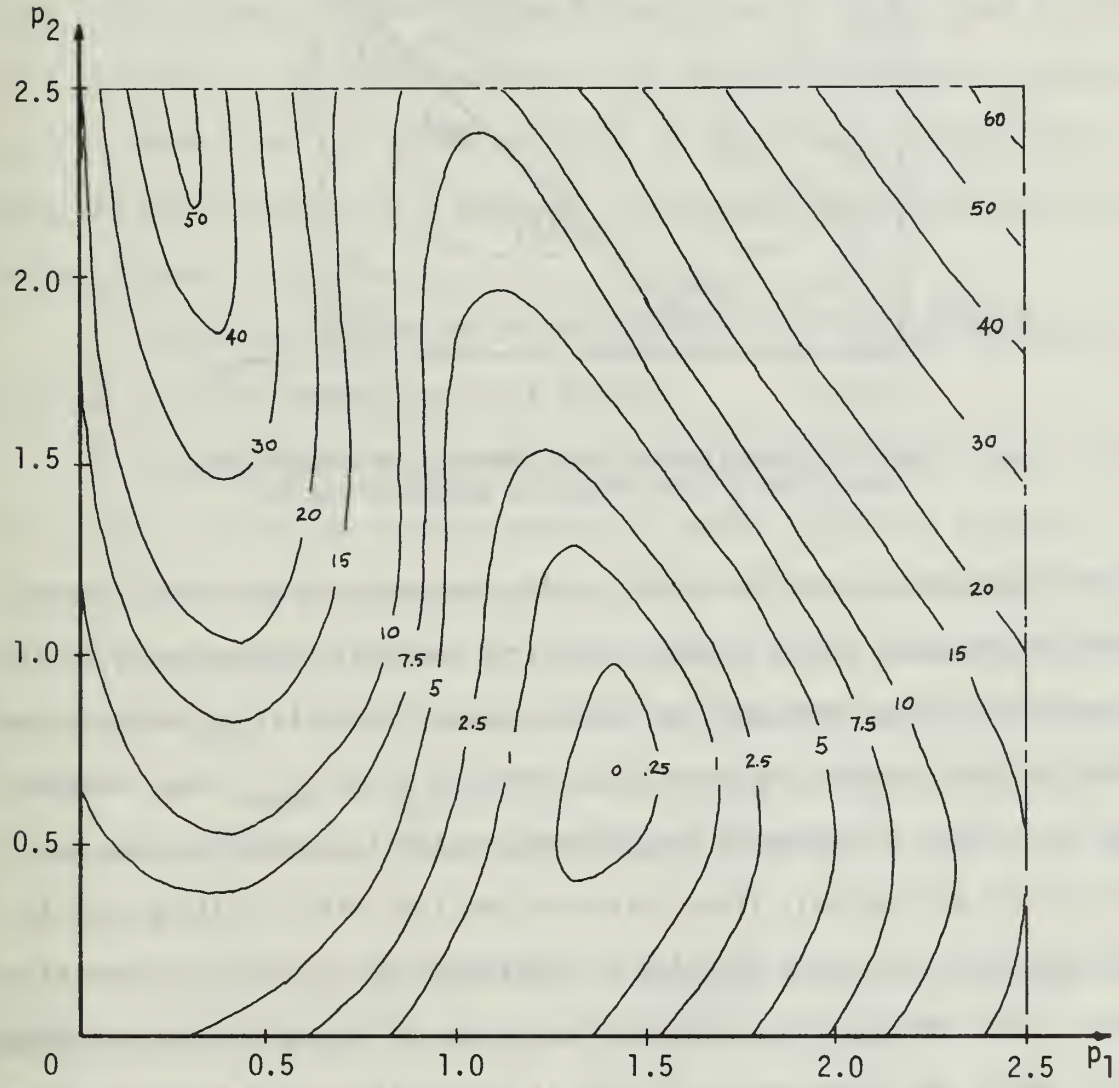
Goal 2. Make efficient use of computer time.

Goal 3. Increase the likelihood of converging to the absolute minimum, if local minima should be present.

Investigation of various possible methods produced one that showed excellent promise of fulfilling all three requirements. This interpolation does not use successive values of the gradient vector, but successive values of the cost function. A simple example is the easiest means of



(a)



(b)

Figure 3

Actual Contour Map Showing Lines of Equal Value of $J(\underset{\sim}{p})$ as a Function of the Two Variable Elements

that the scheme was not functioning so that ever-decreasing values of the cost function were obtained. This helped confirm what was already suspected: the cost function, J(p), as applied to filter network frequency response may not be a convex function. An actual contour map of J(p) as a function of two variable parameters is shown in Figures 3(a) and (b). This map was produced by solving for the cost function over a grid of values of the two variables and joining points of equal cost-function value. The map shows that while only one minimum exists, the function J(p) is obviously not convex. Higher-dimensional maps cannot be easily drawn, but experience in minimizing filters with several variable elements indicates that more than one minimum may be present. Therefore it is concluded that the cost function, J(p), as used in this procedure is not convex, and is possibly, in general, not unimodal. No conclusive results in this area were sought, as this was not the object of the investigation.

5. The Function-Search Interpolation Procedure

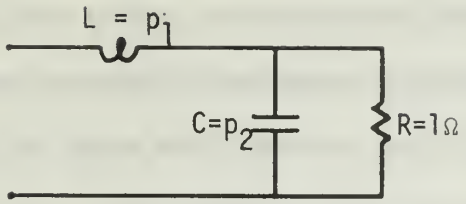
Armed with the knowledge of the cost function's complexity, a new interpolation method was sought. This method would ideally achieve the following goals:

Goal 1. Continually decrease the value of the cost function, never accepting any new point unless it yields a lower value of J(p).

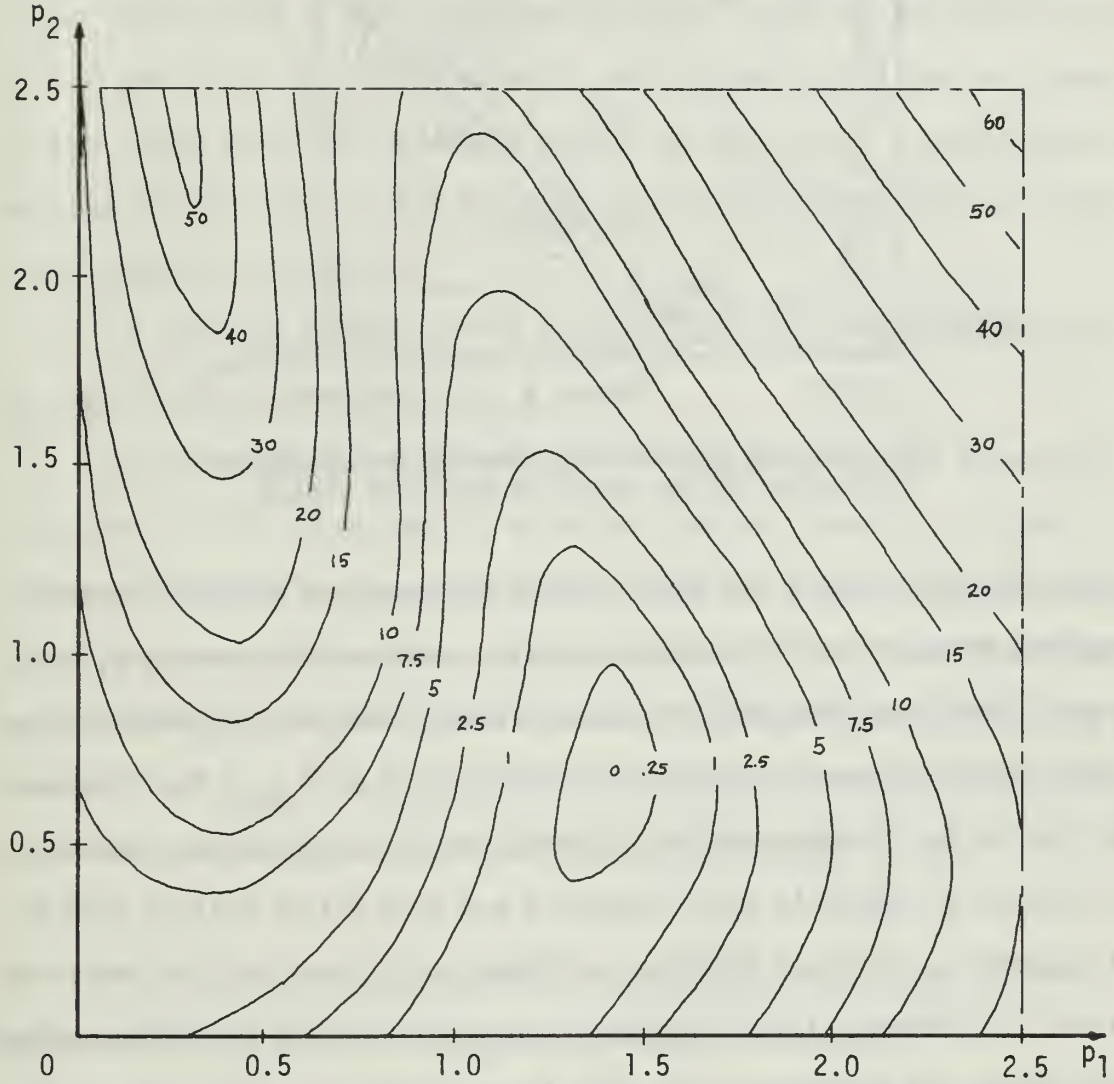
Goal 2. Make efficient use of computer time.

Goal 3. Increase the likelihood of converging to the absolute minimum, if local minima should be present.

Investigation of various possible methods produced one that showed excellent promise of fulfilling all three requirements. This interpolation does not use successive values of the gradient vector, but successive values of the cost function. A simple example is the easiest means of



(a)



(b)

Figure 3

Actual Contour Map Showing Lines of Equal Value of $J(\underset{\sim}{p})$ as a Function of the Two Variable Elements

explaining the interpolation procedure. Using a filter with two variable elements (so that geometrical representations are possible), a contour map of $J(\underset{\sim}{p})$ as a function of the variables might be as shown in Figure 4.

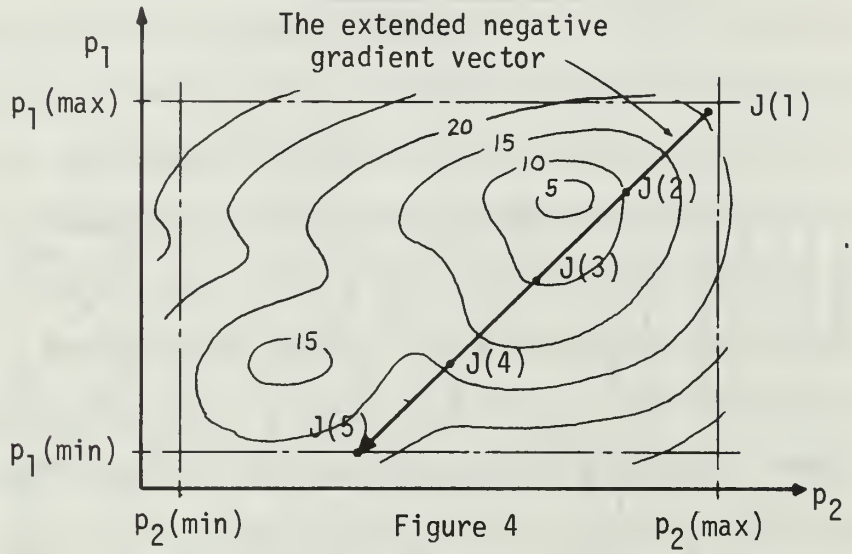


Figure 4

Hypothetical Contour Map Showing the Segmented Extension of the Negative Gradient Vector

At the base point where the cost is J(1), the negative gradient is calculated and extended until it meets the first constraint boundary, p_1(min), as would have been done with the inner product interpolation method. The length of the extended vector will be referred to as T_{max}. Now, however, the line in the direction of the extended vector is subdivided into an even number of intervals (four intervals and five points will be used in this example). The cost function is determined for each of the remaining points, J(2) through J(5). The smallest value of J and its corresponding point number are determined. If J(5) is the smallest cost-function value, no interpolation is performed; if J(1) is the smallest, the interval between J(1) and J(2) is investigated, since the gradient vector's direction indicates that some point with a smaller value than J(1) lies in this direction. If an interior point is smallest, interpolation must be

accomplished. As is seen in Figure 4, J(2) has the smallest cost-function value. The interpolation scheme then passes a parabola through J(2) and its adjacent points (J(1) and J(3)), determines the minimum point of this parabola along the line in the direction of the extended vector, T, and evaluates the cost function, J(6), at this new point, as shown in Figure 5(a). The purpose of approximating the actual curve of the cost function by a parabola, is that for reasonably well-behaved functions, a parabola joining three relatively close points on the curve will closely approximate the actual curve. This helps to increase the speed with which the actual minimum is found.

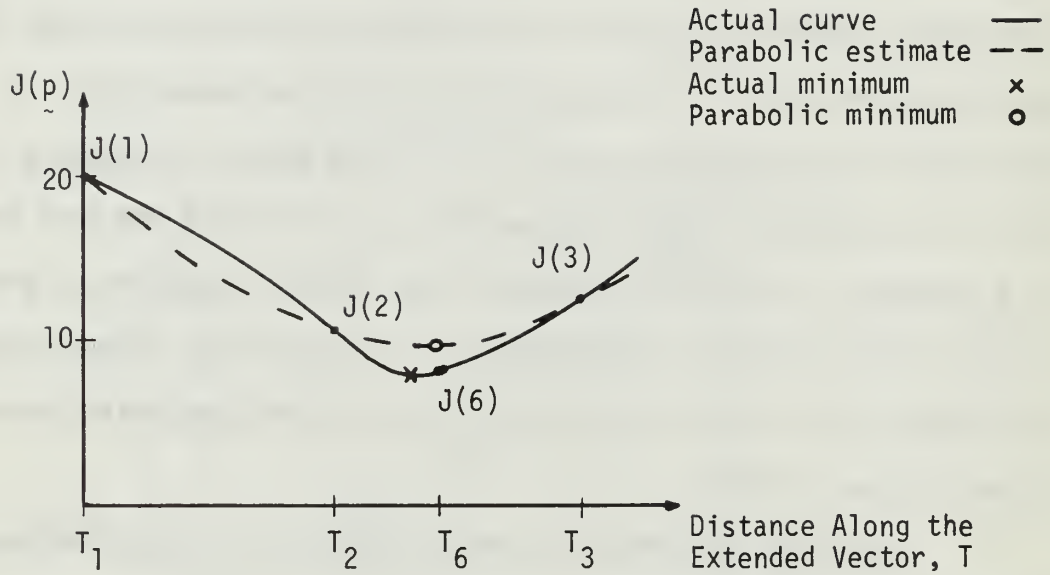
A detailed explanation of the procedure for finding the parabolic minimum is contained in Appendix A.

Having solved for J(6) as shown in Figure 5(a), the points J(2), J(6), and J(3) are then used to solve for another parabolic minimum (Figure 5(b)). This process continues until the difference in extended vector length of the location of two successive minima is within five percent of T_{max}. In this case, location of parabolic minima would terminate if $T_7 - T_6 = 0.05\ T_{max}$. The interpolation is then considered complete, and the lowest cost function value achieved is used as the new starting point.

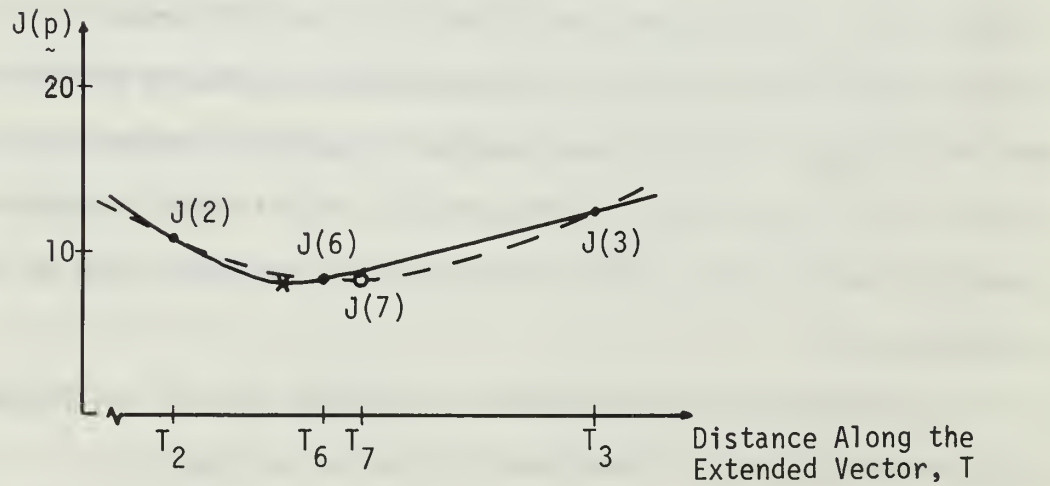
In numerous examples, this interpolation process has succeeded in quite reasonably accomplishing each of the desired goals.

Goal 1. The requirement to continually decrease the value of the cost function was achieved.

Goal 2. Computation time has been reduced over the inner-product interpolation method. Approximately the same number of points along the gradient vector are necessary in the function-search method, but only the value of the cost function, and not the gradient vector, is required at



(a)



(b)

Figure 5

Parabolic Estimation Diagrams

each point. This eliminates n(m - 2) cost-function calculations, where n is the number of variables, and m is the number of points considered along the projection vector (the function search method still requires the gradient vector at the first and last points, accounting for the -2).

Goal 3. The likelihood of converging to the absolute minimum appears to have been increased by use of this method. Results from its use in circuit frequency-response minimization indicate that such may be the case. Appendix B cites two examples (many more have been noted) where a distinct "break" occurred in what appeared to be the final approach stages where the gradient procedure converges to a minimum point. A sudden and reasonably large change in parameter values and decrease in cost-function value take place, something that does not occur using gradient methods that do not make use of the extended vector.

IV. RESULTS

A general result is first presented, followed by specific examples illustrating results for several of the design problems investigated. The final section of the chapter discusses a problem encountered during the investigation.

A. GENERAL RESULTS

Since only a limited number of filters were investigated, the results are not conclusive but serve to demonstrate that optimization of the frequency response of a given filter is feasible using the techniques previously described. Although all the examples are of the passive variety, use of active elements should be feasible using models for the active elements required for the Calahan subroutines described in Reference 3.

B. SPECIFIC RESULTS

In this section, specific results for those filters investigated that representatively demonstrate the capability of the optimization program are presented with circuit diagrams, program solutions and graphical results.

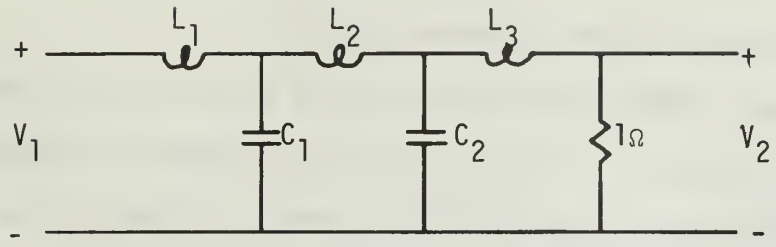
1. Example 1

Figure 6 contains the results of the optimization of a six-element low-pass filter. The desired curve is the Butterworth filter response [Ref. 1]:

$$|\text{Gain}| = 20 \text{ Log}_{10} [(1 + \omega^{2n})^{-1/2}] \qquad (4.1)$$

where n is the number of reactive elements. In this case, n = 5. The actual response deviates from the desired only slightly. A constant

Circuit Configuration



Starting Values of the Variable Parameters

$L_1 = 1.0$ $L_2 = 1.1$ $L_3 = 1.0$ $C_1 = 2.4$ $C_2 = 0.9$

Final Values of the Variable Parameters

$L_1 = 1.557$ $L_2 = 1.695$ $L_3 = .500$ $C_1 = 1.478$ $C_2 = 1.064$

Number of Iterations — 43 Computer Run-time — 136 seconds

Termination Caused by $J^{(i)} - J^{(i+1)} < 10^{-9}$

Graphical Comparison of Desired and Actual Frequency Response

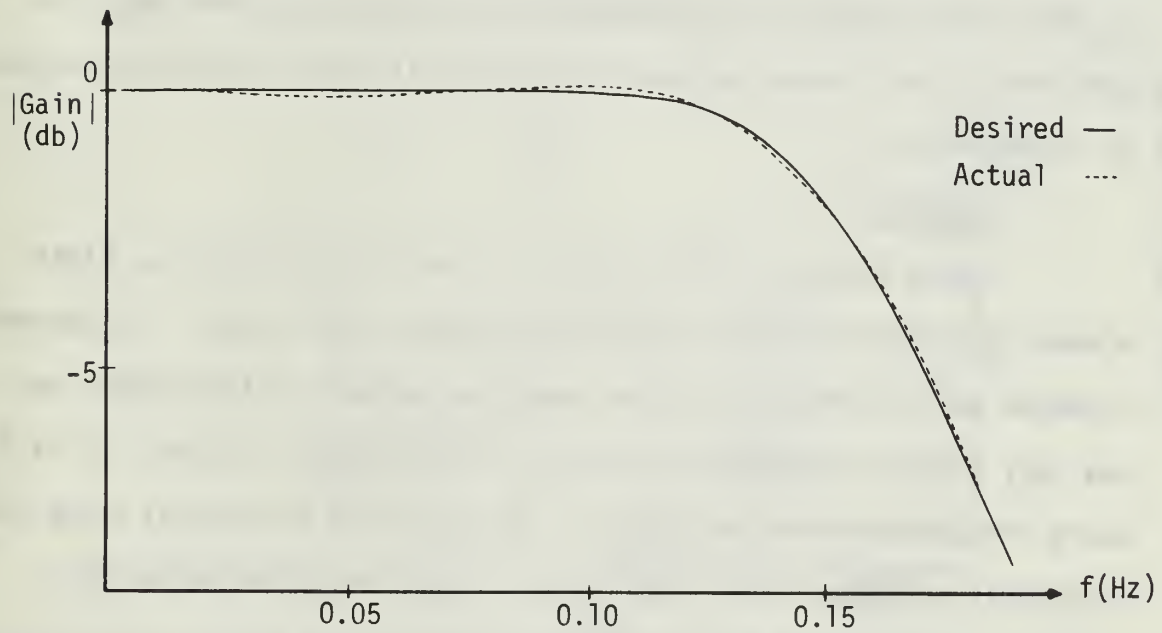


Figure 6

Data for Example 1

weighting matrix ($h_i = 1., i = 1,...,K$) was used in this example, and in all other examples unless otherwise stated.

Figure 7 demonstrates the rapidity with which the program improves the actual response. The filter response is shown in comparison to the desired response after the indicated number of iterations for the problem described in Figure 6. After ten iterations the actual response is a relatively good approximation to the desired response.

2. Example 2

Figure 8 contains the results of the optimization of a five-element low-pass filter, using straight-line approximation for the desired frequency response. Since this is not a realizeable response, it is only possible to approximate it with the actual response. The results are reasonably close to the desired shape and bandwidth. If it were desired to more closely match the 3-db bandwidth, a weighting matrix could be used that places greater emphasis on the points where the desired response is -3 decibels.

3. Example 3

Figure 9 contains the results of the optimization of an eight-element band-pass filter, wherein six elements were varied. The desired response was realizeable with the specified network configuration, and was very closely approximated, except in the frequency interval (0.14,0.22) where the deviation was noticeable. The bandwidth and general shape are reasonably accurate. This same filter circuit was optimized using a weighting matrix where both the upper and lower 3-db points were weighted with a value five times that of the weight of the other points. The deviation at the 3-db points was improved by a factor of about three.

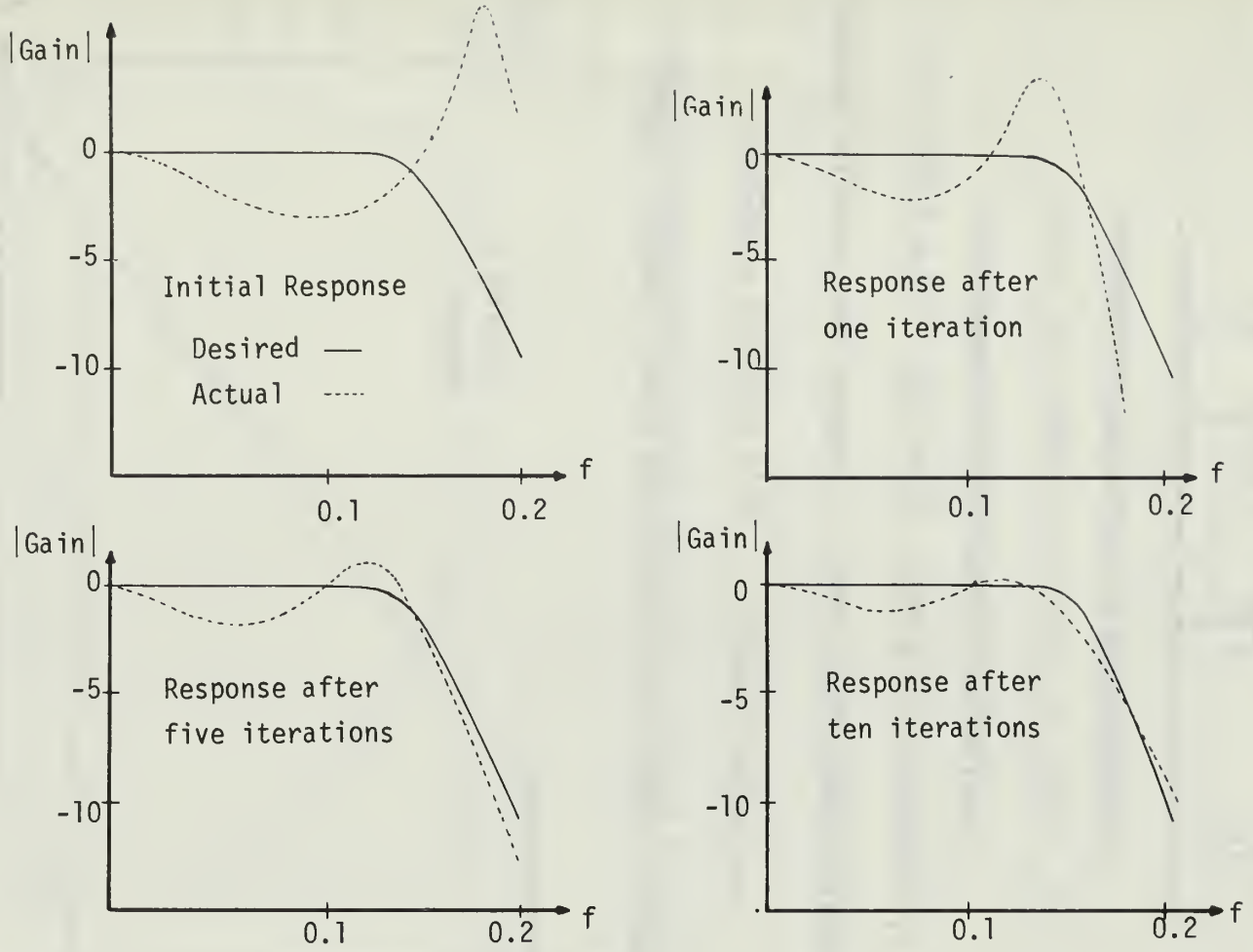
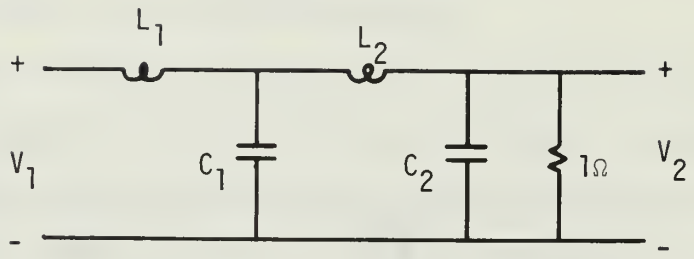


Figure 7

Graphs Showing Sequential Improvement in Actual Response

Circuit Configuration



Starting Values of the Variable Parameters

$L_1 = 2.0$ $L_2 = 2.0$ $C_1 = 2.0$ $C_2 = 2.0$

Final Values of the Variable Parameters

$L_1 = 1.434$ $L_2 = 1.093$ $C_1 = 1.635$ $C_2 = 0.385$

Number of Iterations — 40 Computer Run-time — 132 seconds

Termination Caused by $J^{(i)} - J^{(i+1)} < 10^{-6}$

Graphical Comparison of Desired and Actual Frequency Response

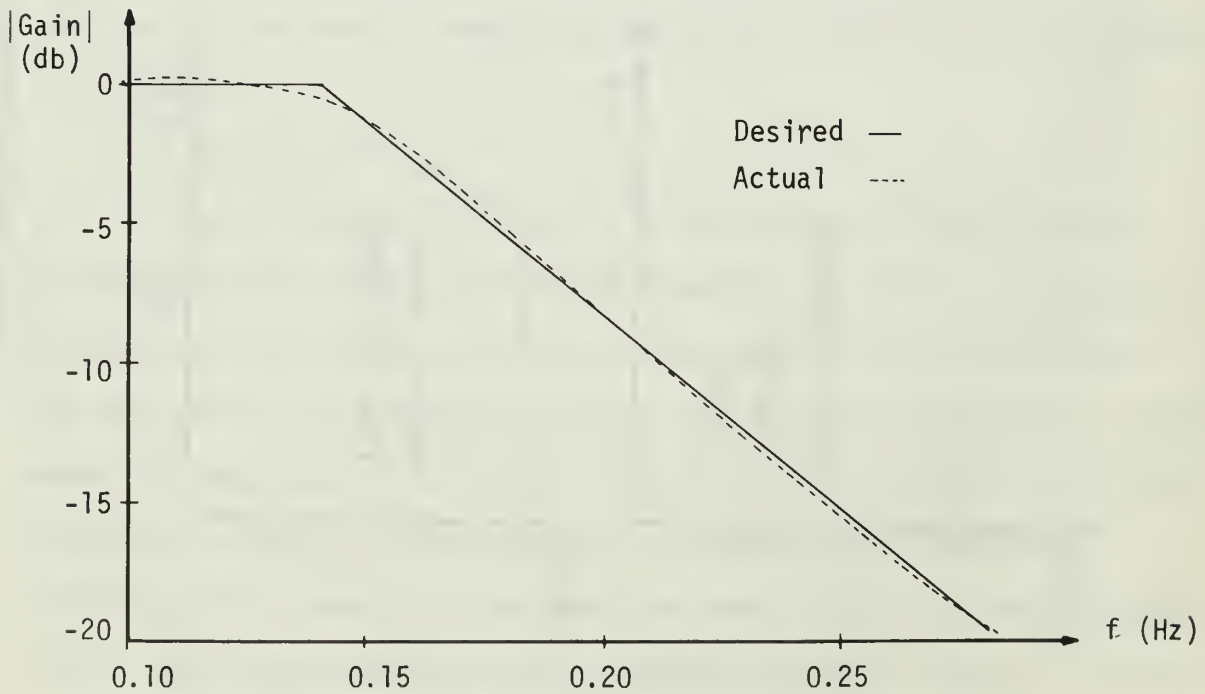
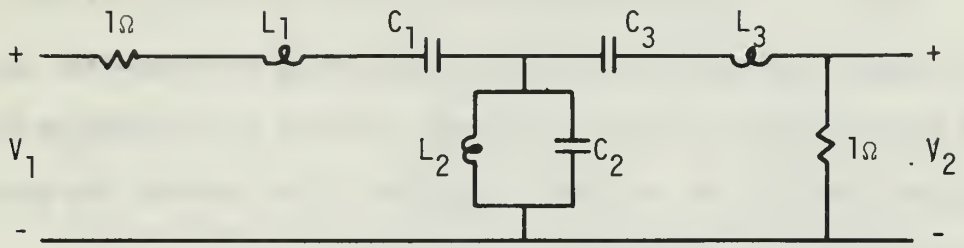


Figure 8

Data for Example 2

Circuit Configuration



Starting Values of the Variable Parameters

$L_1 = 1.0$ $L_2 = 1.0$ $L_3 = 1.0$ $C_1 = 1.2$ $C_2 = 1.0$ $C_3 = 1.2$

Final Values of the Variable Parameters

$L_1 = 0.231$ $L_2 = 0.520$ $L_3 = 0.237$ $C_1 = 1.140$ $C_2 = 0.485$ $C_3 = 1.140$

Number of Iterations — 58 Computer Run-time — 328 seconds

Termination Caused by $J^{(i)} - J^{(i+1)} < 10^{-6}$

Graphical Comparison of Desired and Actual Frequency Response

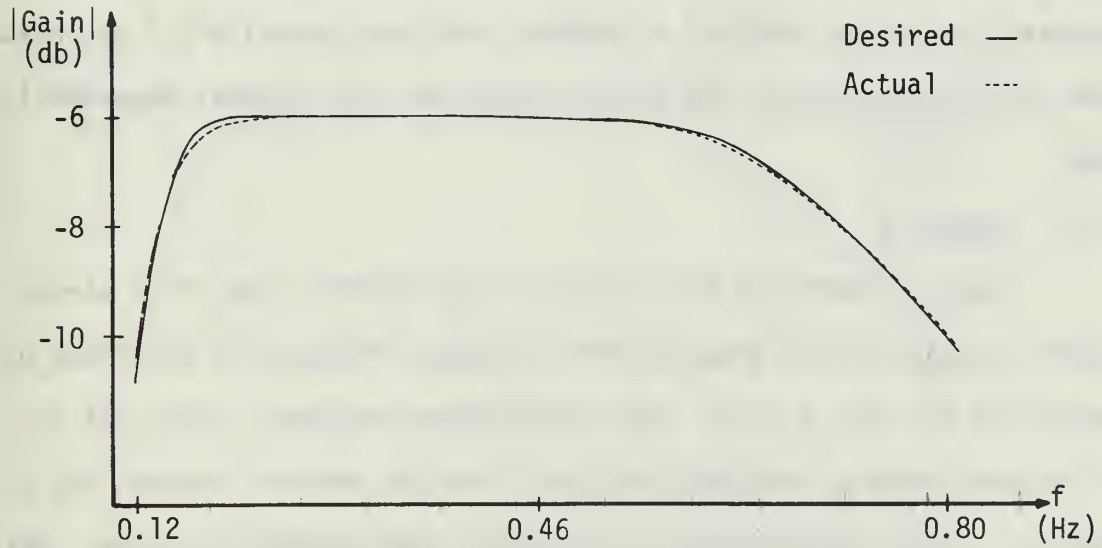


Figure 9

Data for Example 3

4. Example 4

Figure 10 contains the results of the optimization of the same circuit used in example 3, this time specifying a straight-line approximation to the desired frequency response. Again, an approximate solution is the best that can be achieved. The final curve matches the desired reasonably well in general shape and bandwidth.

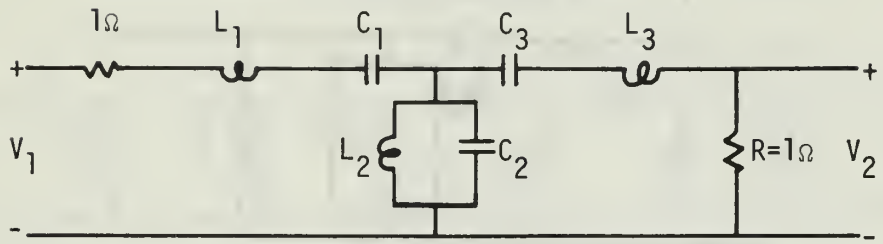
5. Example 5

Figure 11 contains the results of the optimization of a five-element band-pass filter where non-ideal inductors are considered. The desired frequency response is realizeable with the given circuit configuration using ideal elements. When a resistance of $R_s = 0.1$ is included in series with both inductors, the optimized response is slightly lower in gain than desired, but still is a good approximation as shown in the graphical comparison. The same circuit was optimized with $R_s = 0.01$ and with $R_s = 0.5$. The former gave virtually an exact match to the desired response; the latter produced a response that was generally 3.5 decibels below the flat portion of the desired response, and somewhat degraded in shape.

6. Example 6

Figure 12 contains the results of the optimization of an eleven-element low-pass filter whose desired frequency response is described by equation (4.1), with $n = 10$. The optimization produced a close fit to the desired response (maximum deviation from the desired response was 0.27 decibels), but at the expense of relatively long computer run-time. This example required slightly over 32 minutes and 99 iterations to produce the final solution. After 60 iterations, the same general response had been found, with a maximum deviation of 0.42 decibels (and an approximate

Circuit Configuration



Starting Values of the Variable Parameters

$L_1 = 1.0$ $L_2 = 1.0$ $L_3 = 1.0$ $C_1 = 1.2$ $C_2 = 1.0$ $C_3 = 1.2$

Final Values of the Variable Parameters

$L_1 = 0.187$ $L_2 = 0.492$ $L_3 = 0.316$ $C_1 = 1.160$ $C_2 = 0.492$ $C_3 = 1.100$

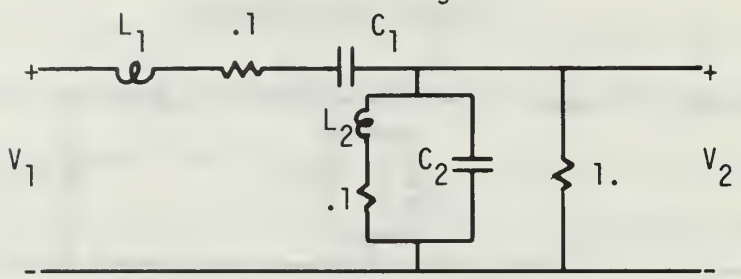
Number of Iterations — 110 Computer Run-time — 670 seconds

Termination Caused by $J^{(i)} - J^{(i+1)} < 10^{-5}$

Graphical Comparison of Desired and Actual Frequency Response

Figure 10
Data for Example 4

Circuit Configuration



Starting Values of the Variable Parameters

$L_1 = 1.0 \quad L_2 = 1.0 \quad C_1 = 1.0 \quad C_2 = 1.0$

Final Values of the Variable Parameters

$L_1 = 0.301 \quad L_2 = 0.745 \quad C_1 = 0.877 \quad C_2 = 0.345$

Number of iterations — 48 Computer Run-time — 187 seconds

Termination caused by $J^{(i)} - J^{(i+1)} < 10^{-7}$

Graphical Comparison of Desired and Actual Frequency Response

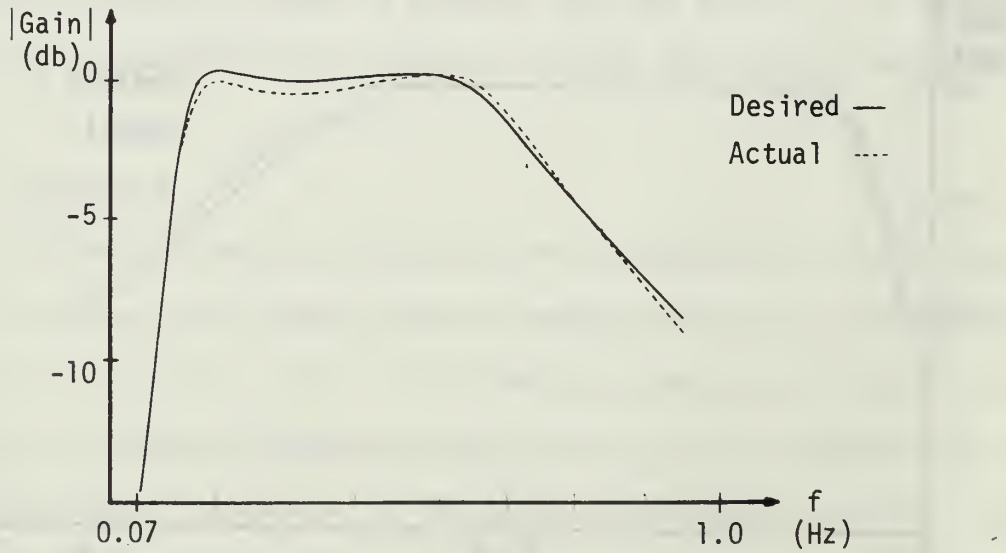
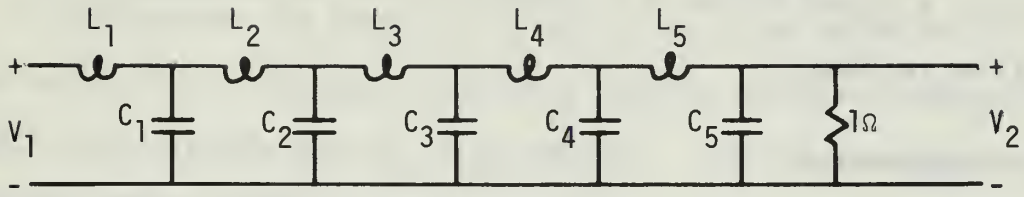


Figure 11
Data for Example 5

Circuit Configuration



Starting Values of the Variable Parameters

$L_1 = 2.$ $L_2 = 2.$ $L_3 = 1.$ $L_5 = 1.$ $C_1 = 2.$ $C_2 = 2.$ $C_3 = 1.$ $C_4 = 1.$ $C_5 = 1.$

Final Values of the Variable Parameters

$L_1 = 1.551$ $L_2 = 1.762$ $L_3 = 1.452$ $L_4 = 1.043$ $L_5 = 0.697$

$C_1 = 1.889$ $C_2 = 1.750$ $C_3 = 1.355$ $C_4 = 0.948$ $C_5 = 0.374$

Number of Iterations — 99 Computer Run-time — 1930 seconds

Termination caused by $J^{(i)} - J^{(i+1)} < 10^{-6}$

Graphical Comparison of Desired and Actual Frequency Response

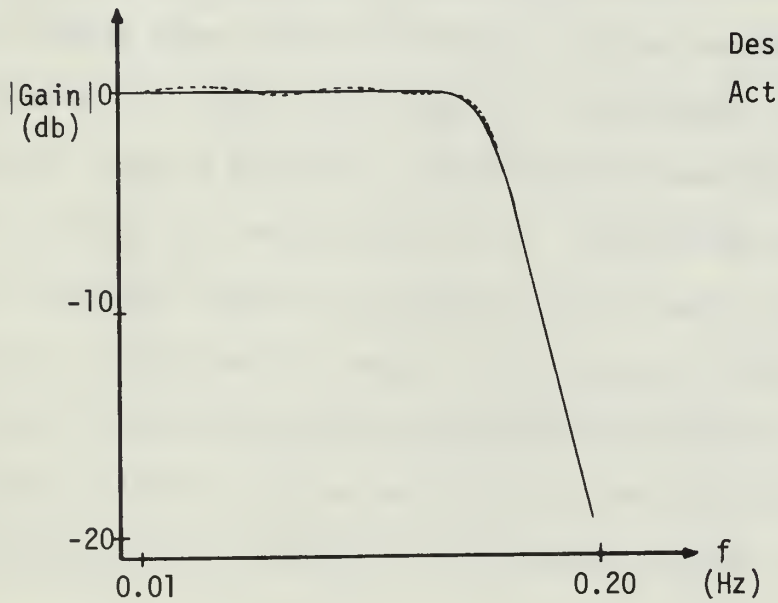


Figure 12

Data for Example 6

run-time of nineteen minutes). This demonstrates the price paid to achieve a higher degree of "closeness of fit" of the actual to the desired response.

C. PROBLEM AREA

In attempting to determine the maximum capability of the program, a complex sixteen-element low-pass filter was attempted. Results showed an unusually long computation time for one iteration (approximately 200 seconds per iteration). Investigation disclosed that about eighty percent of the computational time was spent in the tree-finding subroutine of the modified Calahan section. As a result, after using 720 seconds of computer run-time and obtaining only 3.25 iterations, the problem was abandoned as requiring excessive computation time for a reasonable solution.

V. A GUIDE TO THE USE OF THE OPTIMIZATION PROGRAM

In this section the general characteristics of the optimization program are first discussed, followed by examples which are explained in detail and illustrate the use of the program. Finally, several guidelines are offered that result from experience with the program.

A. GENERAL CHARACTERISTICS

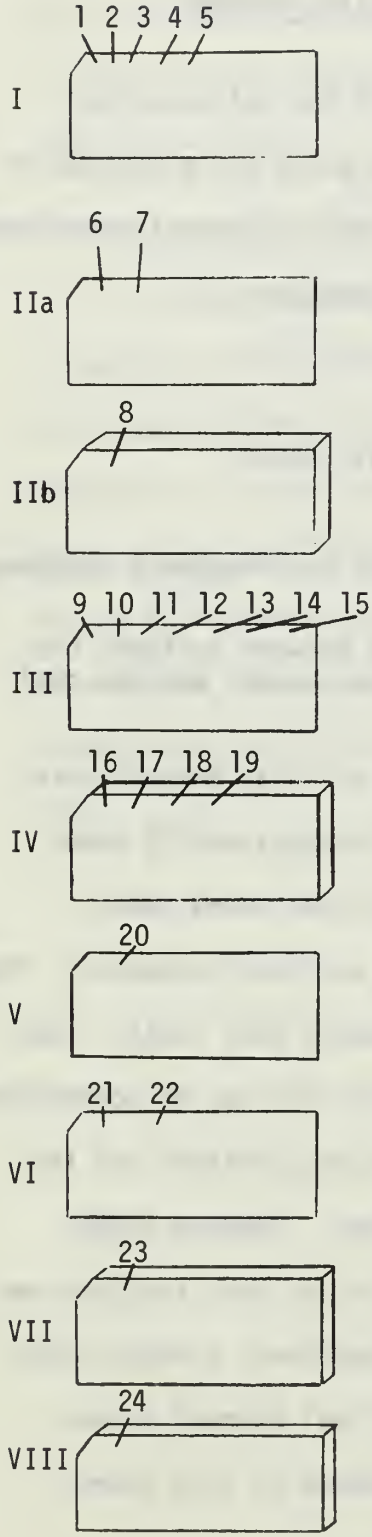
The optimization program consists of three main parts:

1. The gradient-projection search section
2. The modified Calahan program, which produces the frequency response of the given network.
3. Subroutine GRAD which acts as the interface between sections one and two, and computes values for the gradient vector and the cost function.

The program is dimensioned to accept fifteen variable network parameters in a circuit with as many as 100 passive elements and 30 nodes. The circuit to be optimized is topologically described using nodes, branch element types, and their values. For the variable parameters, the specified value is the "starting point"; for elements that remain fixed the actual value is specified. Provisions are made for use of constraints and a cost-function weighting matrix, $\underset{\sim}{h}$. Termination criteria and the desired frequency response are supplied by the user. Program output includes: final parameter values; final values of the cost function and the norm of the gradient; the total number of iterations; printer plots of frequency response, phase, and delay for the final element values. Total storage space required for the computer program is 120K bytes.

B. EXAMPLES

Figure 13 is a general explanation of the required and the optional data cards necessary for use of the program. The specific format for each



I 1. NVAR — the number of elements to be varied

2. NCON — if NCON is set to 0, Card IIa is used to designate common upper and lower bounds; if NCON is set to the number of constraint equations, Card group IIb is used, and Card IIa is omitted.

3. KWIT — Maximum number of iterations desired

4. EPS4 — ε of equation (2.3)

5. WM(1) — h_1 in the weighting matrix, and indicates that Card group VIII will be provided; if h_1 is set equal to 0, equal weighting of all frequency response points will be used by the program

IIa 6. The common lower constraint

7. The common upper constraint

IIb 8. This block of cards designates all constants in the constraint matrix, if required

III 9. The number of passive elements

10. The number of active elements

11. The number of nodes

12. The positive input node number

13. The negative input node number

14. The positive output node number

15. The negative output node number

IV This block of cards describes the circuit topology. The first n cards must be the elements to be varied.

16. The "from" node number

17. The "to" node number

18. The element type (R, L, C)

19. The element value

V 20. The number of frequency intervals

VI 21. The minimum frequency (Hz)

22. The maximum frequency (Hz)

VII 23. This block of cards describes the desired frequency response in decibels

VIII 24. This block of cards is required only if the user desires to specify the weighting matrix, and is necessary if item (5.) is greater than 0

Figure 13

Data Cards Necessary for Use of the Optimization Program

card (or group of cards), is shown in the two examples that follow. It should be noted that two of the entries in Card I serve dual functions.

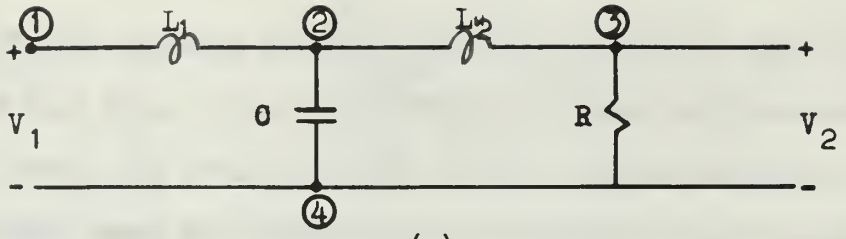
1. If the second entry (NCON, which indicates the number of constraint equations) is set equal to zero, upper and lower bounds for all variable elements are specified with Card IIa, satisfying the equation $a \leq p_i \leq b$, $i = 1,\ldots,n$ where a is the common lower bound and b is the common upper bound; if a positive integer is entered, Card group IIb must be provided by the user, specifying the constants in the constraint equations, omitting Card IIa. (The formation of the constraint matrix is discussed in subparagraph B.2. of this chapter.)

2. If the fifth entry in Card I is set equal to zero, all frequency-response points will be weighted equally, and Card group VIII is omitted; if any positive number other than zero is entered, the program stores this value as the first element of the weighting matrix, h_1. Card group VIII (containing K-1 cards) representing $h_2,\ldots,h_K$ must then be provided as the last section of the data deck.

Two examples will be given using a four-element filter. The first example demonstrates use of the program in the simplest fashion, while the second illustrates the use of more complex constraint and weighting matrices.

1. Example 1

The circuit to be optimized is shown in Figure 14(a). The load resistor, R, is to be held constant; all other elements are to be optimized to match a given frequency response. A maximum of fifty iterations and $\varepsilon = 10^{-6}$ (see equation (2.3)) are the termination criteria. All frequency points are to be equally weighted; eleven frequency points will be matched over the frequency interval (0.10,0.20). The three variable



(a)

CARD NUMBER	DATA	FORMAT
1	3 0 50 .000001 0.	(3I4,2E15.8)
2	0.1 4.	(2F10.0)
3	4 0 4 1 4 3 4	(7(I2,1X))
4	1 2 L 2.0	(2(I2,1X),A1,1X,F10.0)
5	2 4 C 1.5	
6	2 3 L 1.5	
7	3 4 R 1.5	
8	10	(I3)
9	0.10 0.20	(2E15.8)
10	-0.005018	(E15.7)
⋮	⋮	
20	-10.78127	

(b)

Figure 14

Data Cards for Example 1

parameters are to be constrained by the inequality $0.1 \leq p_i \leq 4.0$, $i = 1,2,3$. Figure 14(b) lists the required data cards. An explanation of the information on the cards follows.

Card 1. In sequence:

- 3 - The number of variables, n.
- 0 - The number of constraints; in this case, 0 signifies that all variables are limited by like constraints, the values of which are specified on the next card.
- 50 - The maximum number of iterations. The program will terminate with normal output after the 50^{th} iteration, if no other termination criteria has been reached.
- 0.000001 - This designates ε.
- 0. - Assignment of zero to this field indicates that all frequency-response points are to be weighted equally; that is, $h_i = 1.$, $i = 1,\ldots,K$.

Card 2. This designates the lower constraint as 0.1 and the upper constraint as 4.0 for all variables.

Card 3. In sequence:

- 4 - The total number of passive circuit elements
- 0 - The total number of active circuit elements
- 4 - The number of nodes
- 1 - The positive input node number
- 4 - The negative input node number
- 3 - The positive output node number
- 4 - The negative output node number

Cards 4-7. These cards describe each branch element, as specified earlier. Note that the card for the non-variable load resistor follows the cards for the three variable elements.

Card 8. The number of frequency <u>intervals</u> (the number of frequency points to be matched minus one) is entered.

Card 9. The lower and upper frequency limits are entered.

Cards 10-20. These cards list the desired frequency-response values at equal intervals, from the lowest frequency to the highest. There must be K points specified. The values must be in decibels. A maximum of 100 frequency points may be entered.

CARD NUMBER	DATA	FORMAT
1	3 7 100 .000001 1.	(3I4,2E15.8)
2	1.	(E18.8)
3	0.	
4	0.	
5	1.	
⋮	⋮	
22	1.	
23	.1	
⋮	⋮	
29	0.	
30	4 0 4 1 4 3 4	(7(I2,1X))
31	1 2 L 2.0	(2(I2,1X),A1,1X,F10.0)
32	2 4 C 1.5	
33	2 3 L 1.5	
34	3 4 R 1.0	
35	10	(I3)
36	.1 .2	(2E15.8)
37	-.005018	(E15.7)
⋮	⋮	
47	-10.78127	
48	2.	(E15.8)
49	3.	
⋮	⋮	
57	1.	

Figure 15

Data Cards for Example 2

2. Example 2

The same circuit is to be optimized, in this case using specific constraint and weighting matrices. The constraints are:

$$0.1 \leq p_1 \leq 1.5$$

$$0.5 \leq p_2 \leq 2.0$$

$$1.0 \leq p_3 \leq 3.0$$

$$2p_2 \leq p_3$$

These equations must be put into the form

$$\underset{\sim\sim}{A}\underset{\sim}{p} - \underset{\sim}{B} \geq \underset{\sim}{0} \quad ^{*} \tag{5.1}$$

Proper manipulation of the equations yields the seven constraint inequalities shown below in matrix form.

$$\begin{bmatrix} 1. & 0. & 0. \\ 0. & 1. & 0. \\ 0. & 0. & 1. \\ -1. & 0. & 0. \\ 0. & -1. & 0. \\ 0. & 0. & -1. \\ 0. & -2. & 1. \end{bmatrix} \underset{\sim}{p} - \begin{bmatrix} 0.1 \\ 0.5 \\ 1.0 \\ -1.5 \\ -2.0 \\ -3.0 \\ 0. \end{bmatrix} \geq \underset{\sim}{0}$$

The weighting matrix to be used is

$$\underset{\sim}{h} = [1. \quad 2. \quad 3. \quad 4. \quad 5. \quad 6. \quad 5. \quad 4. \quad 3. \quad 2. \quad 1.]$$

This weighting puts greater emphasis on a close fit at the center portion of the frequency response curve. The number of elements in the weighting matrix must correspond to the number of frequency points, K. Figure 15 lists the required data cards. An explanation of the information on the cards follows.

---

*This notation means that each component of the vector $\underset{\sim\sim}{A}\underset{\sim}{p} - \underset{\sim}{B}$ is non-negative.

Card 1. In sequence:

3 - The number of variables, n.

7 - The number of constraints. This corresponds to the number of rows in $\underset{\sim}{A}$, and cannot exceed 32 without re-dimensioning the program.

100 - The maximum number of iterations.

0.000001 - This designates ε.

1. - This assigns a positive value other than zero to the first element of the weighting matrix, and indicates that specified weighting matrix values will be provided.

Cards 2-22. These cards list the values of each element of the constraint matrix $\underset{\sim}{A}$, sequentially by row.

Cards 23-29. These cards list each value of the constraint matrix B.

Cards 30-47. No change from example 1.

Cards 48-57. These cards list the ten remaining values of the elements of $\underset{\sim}{h}$.

C. GUIDELINES

Experience permits the suggestion of several guidelines that may be helpful in using the program.

1. Termination Criteria

Appropriate values for the maximum number of iterations and for ε, depend primarily on the closeness of fit desired. Higher values for the maximum number of iterations (100 or higher), coupled with lower values of ε (10^{-6} or smaller) will ensure a closer fit, but at the expense of computer run-time.

Additionally, the value for γ in equation (2.4)

$$\left\| \frac{\partial J(\underset{\sim}{p})}{\partial \underset{\sim}{p}} \right\| < \gamma$$

is designated within the main program as "ECON" and is set at a value of 10^{-4}. This termination criterion may be altered by the user if desired.

2. Closeness of Fit at Termination

Closeness of fit of the final frequency response to the desired response is somewhat dependent on how closely the circuit is capable of approximating the desired frequency response. If a close enough fit is not achieved, the following steps are recommended.

Step 1. Impose more stringent termination criteria and re-initialize with the final values just obtained. (This step should be omitted if it is clear that the circuit configuration is not capable of coming close to the desired response.)

Step 2. Try different starting points for the variables if the termination value of the cost function is high.

Step 3. Try a different network configuration.

In addition, the value N in equation (2.2), the cost-function equation, has been set at N = 2 for least-squares curve fitting. This may be changed to N = 4 to penalize more heavily large deviations from the desired response. Statements 99 and 101 in subroutine GRAD both contain "**2" which should be changed to "**4" to accomplish this alteration.

3. Number of Frequency Points

Enough frequency points should be specified to completely describe the desired response. Doubling the number of frequency-response points does not double the computer run-time, but requires an increase of only about five to twenty percent in time (the larger percentage applies to small circuits, the smaller percentage to large circuits). This is because most of the run-time used is spent producing the network function (poles and zeros); once this is done, computation of the frequency

response is relatively fast. Failure to adequately describe the desired response may produce poor results. Experience with a limited number of examples indicates that an adequate description is provided when frequency points are provided to at least ten decibels below the desired flat response for low- and band-pass filters, and spaced over a close enough interval so that "corners" are well-defined.

4. Computation Time

Complex circuits (those with many trees) and any circuit with more than about eight elements require run-time in excess of ten minutes (on the IBM-360 digital computer). Investigation has shown that this time requirement is a function that varies directly with the number of trees in the circuit. Appropriate alteration of the Calahan subroutines to improve or alter the tree-finding method should provide a decrease in computation time necessary to produce a reasonable solution for large circuits.

VI. COMPARATIVE RESULTS OF GRADIENT-PROJECTION AND PATTERN-SEARCH METHODS

Optimization of filter frequency response using pattern-search methods is the subject of a Naval Postgraduate School thesis being concurrently written by Captain James Lau, USMC. A comparison of the results of attempts to optimize the frequency response of three networks using each of the programs is shown in Figure 16. The first and second comparisons were optimized to a totally realizeable response, while the third was to a straight-line approximation. In all cases, the pattern-search optimization program proved faster and capable of producing more accurate results.

FILTER TYPE	GRADIENT PROJECTION RESULTS	PATTERN SEARCH RESULTS
1. Eight-element band-pass		
Final cost-function value	0.564×10^{-4}	0.220×10^{-5}
Computation time	328 seconds	184 seconds
2. Five-element low-pass		
Final cost-function value	0.458×10^{-2}	0.204×10^{-5}
Computation time	133 seconds	49 seconds
3. Five-element low-pass		
Final cost-function value	1.655	1.651
Computation time	132 seconds	50 seconds

Figure 16

Results of Three Network Optimizations,
Comparing Gradient-Projection and Pattern-Search Methods

VII. CONCLUSIONS

The purpose of this investigation was to determine the feasibility of using minimization techniques to cause a given network configuration to match a desired frequency response as closely as possible. This method was to use the computer's speed coupled to optimization techniques to provide the filter designer with a tool that quickly determines the network element values that most accurately cause the desired frequency response to be reproduced. Two existing programs capable of

1. minimizing a given function, and
2. producing a given network frequency response

were adapted to produce a program capable of accomplishing the above-stated purpose.

The results presented in Chapter IV and the comparisons in Chapter VI lead to the following conclusions:

1. It is feasible to obtain realistic approximations to a given frequency response, using a circuit capable of attaining that response.

2. Satisfactory solutions using gradient-projection techniques can be obtained for filters with fewer than about twelve elements. An unreasonable amount of time may be required in attempting to optimize larger circuits.

3. The pattern-search method for minimization is shown to be superior to the gradient projection method in both time and accuracy, for the examples compared.

Recommendations for logical extensions of this investigation follow:

1. Investigate a means to determine the gradient vector more accurately and more quickly.

2. Alter the analysis program so that the tree-generating procedure is used only when necessary, rather than in each iteration.

APPENDIX A

PROCEDURE FOR FINDING THE PARABOLIC MINIMUM USED IN THE FUNCTION-SEARCH INTERPOLATION METHOD

1. The general equation of a parabola is:

$$Ax^2 + Bx + C = y \quad . \tag{A.1}$$

The unknown coefficients A, B and C that will cause a parabola to pass through any three known points (not in a straight line) can be found by solving three simultaneous equations:

$$Ax_1^2 + Bx_1 + C = y_1 \tag{A.2a}$$

$$Ax_2^2 + Bx_2 + C = y_2 \tag{A.2b}$$

$$Ax_3^2 + Bx_3 + C = y_3 \tag{A.2c}$$

2. These equations can be solved using the matrix equations:

$$\begin{matrix} A \\ B \\ C \end{matrix} = \underset{\sim}{x}^{-1}\underset{\sim}{y} = \begin{matrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{matrix}^{-1} \begin{matrix} y_1 \\ y_2 \\ y_3 \end{matrix} \tag{A.3}$$

3. The minimum of this parabola can be found by setting the first derivative of (A.1) with respect to x equal to zero and solving for x_{min}.

$$\frac{d}{dx}\left(Ax^2 + Bx + C - y \right) = 2Ax_{min} + B = 0$$

$$\therefore \; x_{min} = -\frac{B}{2A}$$

APPENDIX B

FUNCTION-SEARCH INTERPOLATION EXAMPLES

1. Shown below are two examples (many others were noted, but are not included herein) where the gradient-projection minimization process seemed to be approaching a minimum, (no conclusive proof of this is offered, only the indications listed herein) when a distinct change in all of the variables as well as the cost function occurred. In the first four of the five iterations shown, the changes in the parameter values were small, and the cost function was being very slowly reduced. This behavior is characteristic of gradient methods when a minimum has been found (this can also be representative of moving down a "ridge line" or a "narrow valley"). This "jump" would not take place if the gradient method (as opposed to the gradient-projection method) were used. (Only five iterations are shown, as those before and after are not pertinent to the illustration.)

First Example. Five-element (four-variable) low-pass filter

Iteration number	Variable Parameter Values				Cost function
	p_1	p_2	p_3	p_4	
19	1.3311	1.9541	0.9654	0.5633	0.2425
20	1.3284	1.9489	0.9624	0.5627	0.2363
21	1.3523	1.9178	0.9801	0.5493	0.2171
22	1.3468	1.9108	0.9743	0.5489	0.1978
23	1.4730	1.6550	1.0565	0.4367	.02567

Second Example. Six-element (five-variable) low-pass filter

Iteration number	Variable Parameter Values					Cost function
	p_1	p_2	p_3	p_4	p_5	
32	1.7525	1.8321	1.4150	1.0029	0.4440	0.02815
33	1.7493	1.8270	1.4163	1.0041	0.4469	0.02782
34	1.7503	1.8269	1.4185	1.0037	0.4488	0.02733
35	1.7497	1.8263	1.4182	1.0039	0.4503	0.02730
36	1.6674	1.6507	1.5231	1.0256	0.5055	0.01116

2. These examples tend to substantiate the conclusion that the function-search interpolation method used in the gradient projection minimization program increases the likelihood of converging to the absolute minimum, for when the vector in the negative gradient direction intersects a point with a lower cost-function value — perhaps relatively distant from the base point — this new smaller-valued location will be taken up as the new base point.

```
C                        COMPUTER PROGRAM
C
C
C       THIS COMPUTER PROGRAM IS DESIGNED TO MINIMIZE THE DEVIATION
C       BETWEEN ACTUAL AND GIVEN FREQUENCY RESPONSE FOR A GIVEN NETWORK.
C       THE PROGRAM CONSISTS OF THREE SECTIONS:
C
C            (1) THE GRADIENT PROJECTION MINIMIZATION PROGRAM
C
C            (2) THE MODIFIED CALAHAN PROGRAM FOR DETERMINING FREQUENCY
C                RESPONSE
C
C            (3) SUBROUTINE GRAD WHICH ACTS AS AN INTERFACE BETWEEN SECTION
C                (1) AND (2), AND PRODUCES THE VALUE OF THE COST FUNCTION
C                 AND THE DISCRETE APPROXIMATION TO THE GRADIENT VECTOR.
C
C
C                              SECTION (1)
C                 THE GRADIENT PROJECTION MINIMIZATION PROGRAM
C
C
C-----THE INDICES CORRESPONDING TO  ACTIVE CONSTRAINTS ARE IN THE ARRAY
C-----ICON. THE NUMBER OF ACTIVE CONSTRAINTS IS NAC.
C-----FORM THE PROJECTION MATRIX BY USING THE RECURSIVE EQUATIONS
C-----N1T*N1 = 1, THEREFORE THE FIRST INVERSE IS 1
C-----D1W IS A WORKING MATRIX, D1R AND D2R ARE WORKING VECTORS
C-----EHQ CONTAINS ALL OF THE CONSTRAINT INFORMATION
C-----ENQ IS THE UP-TO-DATE NQ MATRIX,VNQ IS A COLUMN OF ENQ
C
C
      DIMENSION MP(100,3),ELT(100),MAP(20,5),ELTA(20),VAL(100),
     1VALA(20),D(100),FTMP(20),WM(100)
      COMMON EHQ(15,32),ENQ(15,15),PQ(15,15),EINV(15,15),
     1D1W(15,15),B(32),D1R(65),D2R(65),Y(30),Z(65),VNQ(30),
     2YTMP(15),GTMP(15),YTRY(15),GTRY(15),DLAM(32),G(15),
     4EPS1,EPS2,EPS3,AZ,DB4,THMIN,HMIN,DN,DZ,DM,DS,DT,RHO,DY,RP,RE,
     5ICON(32),KCON(32),JCON(32),MCON(32),ISAME(30),NVAR,NCON,
     6IDROP,IMAX,NAC,IDEP,NACM1,JV,NMACT,NQP1,NS,NSP1,NVAR2,IND
    1 FORMAT(7(I2,1X)/(I2,1X,I2,1X,A1,1X,F10.0))
    4 FORMAT(4(I2,1X),A1,1X,F10.0)
   43 FORMAT(I3/2E15.8)
   49 FORMAT(E15.7)
  600 FORMAT(5X'A FEASIBLE POINT DOES NOT EXIST'//)
 1000 FORMAT(5X'DEGENERACY 3'//)
 1003 FORMAT(5X'DEGENERACY 1'//)
 1004 FORMAT(5X'DEGENERACY 2'//)
 1006 FORMAT(5X' MINIMUM COST = ' E15.8,5X,'NORM = ',E15.8,5X,'KWIT = ',
     1I6,/)
```

```
 1010 FORMAT(10X'THE OPTIMAL CONTROL HISTORY IS'/, 10(E12.4))
 1011 FORMAT(///,5X'THE OPTIMAL TRAJECTORY IS'/,10(E12.4))
 1012 FORMAT(3I4,2E15.8)
 1017 FORMAT(8(3XE12.4) //)
 1060 FORMAT(E18.8)
 1061 FORMAT(8F10.0)
 160C FORMAT(I2)
      CALL ERRSET(207,300,-1,1,1,210)
      KOUT=0
      LIN=1
      READ(5,1012)NVAR,NCON,KWIT,EPS4,WM(1)
      IF(NCON.EQ.0)GO TO 121
C     READ THE CONSTRAINT MATRIX, IF GIVEN
      READ(3,1060)((EHQ(I,J),I=1,NVAR),J=1,NCON)
      READ(3,1060)(B(I),I=1,NCON)
      GO TO 126
C     FORM THE CONSTRAINT MATRIX FROM THE UPPER AND LOWER BOUNDS, IF
C     DESIGNATED BY READING IN 'NCON = 0'.
  121 NCON=2*NVAR
      READ(5,122)EELO,EEHI
  122 FORMAT(2F10.0)
      NVAP=NVAR+1
      DO 123 I=1,NVAR
      DO 123 J=1,NCON
      EHQ(I,J)=0.
      IF(I.EQ.J) EHQ(I,J)=1.
  123 IF((I+NVAR).EQ.J) EHQ(I,J)=-1.
      DO 124 I=1,NVAR
  124 B(I)=EELO
      DO 125 I=NVAP,NCON
  125 B(I)=-EEHI
  126 EPS1 = .001
      EPS2 = .001
      EPS3 = .001
      ECON = .0001
      FKP=1.E25
C---READ RLC ELEMENTS
      READ(5,1)NPL,NAL,NN,JI,KI,JO,KO,
     1(MP(J,1),MP(J,2),ELT(J),VAL(J),J=1,NPL)
C---READ ACTIVE ELEMENTS, IF ANY
      IF(NAL)3,2,3
    3 READ(3,4)((MAP(J,I),I=1,4),ELTA(J),VALA(J),J=1,NAL)
    2 READ(5,43)NOM,OMGMIN,OMGMAX
      ND=NOM+1
C     READ THE DESIRED FREQUENCY RESPONSE POINTS
      READ(5,49)(D(I),I=1,ND)
      IF(WM(1).EQ.0.)GO TO 128
C     READ THE WEIGHTING MATRIX, IF DESIGNATED BY SETTING WM(1) EQUAL TO
```

```
C       SOME POSITIVE VALUE.
        READ(5,129)(WM(I),I=2,ND)
  129 FORMAT(E15.8)
        GO TO 50
C       FORM THE WEIGHTING MATRIX IF NONE IS SPECIFIED
  128 DO 51 I=1,ND
   51 WM(I)=1.
   50 DO 10 I=1,NVAR
   10 Y(I)=VAL(I)
        DO 220 J=1,NCON
        DN=0.
        DO 221 I=1,NVAR
  221 DN = DN + EHQ(I,J)**2
        DN = SQRT(DN)
        DO 222 I=1,NVAR
  222 EHQ(I,J) = EHQ(I,J)/DN
  220 B(J) = B(J)/DN
        CALL VIOLA
C-----IF J=1, NO CONSTRAINTS ARE VIOLATED, OTHERWISE, ICON CONTAINS THE
C-----CONSTRAINTS WHICH ARE VIOLATED AND D2R CONTAINS THE VALUES OF THE
C-----VIOLATED LAMBDAS.
C-----PRINT THE ARRAYS ICON AND D2R, THEN FIND A FEASIBLE POINT.
C       WRITE(6,1807)JV
 1807 FORMAT(5X,'JV =',I4)
        IF(JV.EQ.0)GO TO 150
  234 EINV(1,1) = 1.
        DO 225 I=1,JV
  225 ICON(I) = KCON(I)
        IQ=ICON(1)
        DO 226 I=1,NVAR
  226 ENQ(I,1) = EHQ(I,IQ)
        DO 227 I=1,NVAR
        DO 227 J=1,NVAR
        PQ(I,J) = -ENQ(I,1)*ENQ(J,1)
        IF(I.EQ.J) PQ(I,J) = PQ(I,J) + 1.
  227 CONTINUE
C-----PQ IS NOW EQUAL TO P1
        NAC = 1
        IF(JV.LT.2) GO TO 231
        DO 230 I=2,JV
        IMAX=ICON(I)
        CALL ADDHY
C-----IF IDEP = 1, A DEPENDENT HYPERPLANE HAS BEEN FOUND
  230 CONTINUE
  231 CONTINUE
        CALL CORREC
C-----CHECK TO SEE IF A FEASIBLE POINT HAS BEEN
   60 HMIN = 1.E5
```

```
      DO 232 J=1,NCON
      D1R(J) = 0.
      DO 233 M=1,NVAR
  233 D1R(J) = D1R(J) + EHQ(M,J)*Y(M)
      D1R(J) = D1R(J) - B(J)
 1810 FORMAT(5X,'D1R(',I3,') =',E14.7)
      IF(D1R(J).GE.HMIN) GO TO 232
      IMAX=J
      HMIN = D1R(J)
  232 CONTINUE
      IF(HMIN.GE.-1.D-10) GO TO 150
C-----IF HMIN.GE.0. ALL CONSTRAINTS ARE SATISFIED, GO TO 150, CHECK TO
C-----SEE WHICH CONSTRAINTS ARE ACTIVE, FORM PQ, CHECK FOR LINEAR
C-----DEPENDENCE AND USE CORRECTION PROCEDURE (P.203) IF REQUIRED.
      DO 235 I=1,NAC
      VNQ(I) = 0.
      DO 235 K=1,NVAR
  235 VNQ(I) = VNQ(I) + ENQ(K,I)*EHQ(K,IMAX)
      DO 236 I=1,NAC
      D2R(I) = 0.
      DO 236 K=1,NAC
  236 D2R(I) = D2R(I) + EINV(I,K)*VNQ(K)
C-----D2R  CONTAINS R OF EQ.(7.7)
      DO 237 I=1,NVAR
      VNQ(I) = EHQ(I,IMAX)
      DO 237 K=1,NAC
  237 VNQ(I) = VNQ(I) - ENQ(I,K)*D2R(K)
C-----VNQ CONTAINS PQ*NIMAX, TEST FOR THE THREE POSSIBILITIES
      DN = 0.
      DO 238 I=1,NVAR
  238 DN = DN + VNQ(I)**2
      IF(DN.GT.EPS1) GO TO 250
      RMAX = 0.
      DO 239 I=1,NAC
      IF(D2R(I).LE.RMAX) GO TO 239
      RMAX = D2R(I)
      IDROP = I
  239 CONTINUE
      IF(RMAX.GT.EPS2) GO TO 240
C-----IF RMAX NOT GREATER THAN 0, A FEASIBLE POINT DOES NOT EXIST
      WRITE(6,600)
      GO TO 550
C-----DROP NIDROP FROM ENQ
  240 CALL DROP
C-----ADD NIMAX TO ENQ
      CALL ADDHY
  252 DO 243 I=1,NVAR
      DO 243 K=1,NAC
```

```
  243 Y(I) = Y(I) - ENQ(I,K)*EINV(K,NAC)*HMIN
C-----NEW Y FORMED AS GIVEN BY EQ.(7.10) OR EQ.(7.13) AS APPROPRIATE
C-----CONSTRAINTS IN ICON ARE NOW SATISFIED
      GO TO 60
  250 CALL ADDHY
      GO TO 252
C-----DETERMINE WHICH CONSTRAINTS ARE ACTIVE AND USE THE PROCEDURE ON
C-----P. 203 TO ACCOUNT FOR LINEAR DEPENDENCE, IF ANY.
  150 DO 151 I=1,NVAR
      DO 151 J=1,NVAR
      PQ(I,J) = 0.
      IF(I.EQ.J) PQ(I,J) = 1.
  151 CONTINUE
C-----DETERMINE WHICH CONSTRAINTS ARE ACTIVE
      CALL ACTIVE
  776 J=0
      NAC = 0
      DO 5 IJK=1,NVAR
    5 VAL(IJK)=Y(IJK)
      CALL GRAD(NPL,NAL,NN,JI,KI,JO,KC,MP,ELT,VAL,LIN,NOM,OMGMIN,NVAR,
     1OMGMAX,D,G,FUNC,1,2,WM)
      CALL FINDZ
  160 CALL THETA
      IF(THMIN.GE.0.) GO TO 161
C-----ADD NIMAX TO NQ
      CALL ADDHY
      J = J + 1
      CALL FINDZ
      IF(J.GT.NMACT)GO TO 490
      IF(J.NE.NMACT) GO TO 160
  161 DS = 0.
      DO 162 I=1,NVAR
  162 DS = DS + Z(I)*G(I)
      IF(DS.LE.0.) GO TO 260
  180 J = 1
C-----Z IS THE PROJECTED GRADIENT VECTOR,DETERMINE THE MAXIMUM STEP
C-----SIZE,TMAX
C     WRITE(6,605)NAC
  605 FORMAT(5X,'DETERMINE MAX STEP SIZE, NAC =',I4)
      JSAME = 0
      TMAX=1.E5
C-----JSAME IS AN INDICATOR -- IF JSAME.GT.0 MORE THAN ONE TAU = TAUM,
C-----USE CORRECTION PROCEDURE
      DO 136 I=1,NCON
      IF(NAC.EQ.0) GO TO 131
      IF(I.NE.MCON(J))GO TO 131
      IF(J.NE.NAC) J=J+1
      GO TO 136
```

```
  131 DN=-B(I)
      DM = C.DO
      DO 132 K=1,NVAR
      DM = DM +EHQ(K,I)* Z(K)
  132 DN = DN +EHQ(K,I)* Y(K)
      IF( ABS(DM).LE.1.D-10) GO TO 136
      T =  -DN/DM
      IF (T.LE.1.D-8)GO TO 136
      IF(T.GT.TMAX)GO TO 133
      IF(TMAX-T.LE.1.D-8) GO TO 134
      JSAME = 0
      GO TO 135
  134 JSAME = JSAME + 1
      ISAME(JSAME) = I
  135 TMAX = T
      IMAX = I
      GO TO 136
  133 IF(T-TMAX.GT.1.D-8) GO TO 136
      JSAME = JSAME + 1
      ISAME(JSAME) = I
  136 CONTINUE
C-----ADD A HYPERPLANE.  ANY LINEARLY DEPENDENT HYPERPLANES
C-----ARE INDICATED IN THE ARRAY ISAME.
C-----DETERMINE IF TAKING THE MAXIMUM STEP IS TOO FAR, THAT IS, IS
C-----INTERPOLATION REQUIRED
      IKP=1
      FTMP(1)=FUNC
      TAU=TMAX
      DO 110 I=1,NVAR
  110 VAL(I)=Y(I)+TMAX*Z(I)
      CALL GRAD(NPL,NAL,NN,JI,KI,JO,KC,MP,ELT,VAL,LIN,NOM,OMGMIN,NVAR,
     1OMGMAX,D,G,F,1,1,WM)
      FTMP(5)=F
  105 DO 100 I=2,4
      DO 101 J=1,NVAR
  101 VAL(J)=Y(J)+TMAX*Z(J)*(I-1)/4.
      CALL GRAD(NPL,NAL,NN,JI,KI,JO,KC,MP,ELT,VAL,LIN,NOM,OMGMIN,NVAR,
     1OMGMAX,D,G,F,1,1,WM)
      FTMP(I)=F
  100 IF(FTMP(I).LE.FTMP(IKP)) IKP=I
      IF(IKP.GT.1)GO TO 104
      TMAX=TMAX/4.
      FTMP(5)=FTMP(2)
      GO TO 105
  104 X3=IKP/4.*TMAX
      X2=X3-.25*TMAX
      X1=X2-.25*TMAX
  113 W1=X1**2
```

```
      W2=X2**2
      W3=X3**2
      FAC=W1*X2+W3*X1+W2*X3-W3*X2-W1*X3-W2*X1
      A=((X2-X3)*FTMP(IKP-1)+(X3-X1)*FTMP(IKP)+(X1-X2)*FTMP(IKP+1))
     1/FAC
      BE=((W3-W2)*FTMP(IKP-1)+(W1-W3)*FTMP(IKP)+(W2-W1)*FTMP(IKP+1))
     1/FAC
      IF(A.LE.C.)GO TO 106
      RHO=-BE/A*.5
      IF(RHO.GE.TAU)GO TO 106
C-----IF A IS NEGATIVE, OR RHO GT TAU, NO INTERPOLATION IS REQ'D
      DO 103 I=1,NVAR
  103 VAL(I)=Y(I)+RHO*Z(I)
      CALL GRAD(NPL,NAL,NN,JI,KI,JO,KO,MP,ELT,VAL,LIN,NOM,OMGMIN,NVAR,
     1OMGMAX,D,G,F,1,1,WM)
      IF(F.GT.FTMP(IKP))GO TO 117
      IF(ABS(RHO-X2).LT..01*TAU)GO TO 107
      IF(RHO.GT.X2)GO TO 112
  116 X3=X2
      X2=RHO
      FTMP(IKP+1)=FTMP(IKP)
      FTMP(IKP)=F
      GO TO 113
  112 X1=X2
      X2=RHO
      FTMP(IKP-1)=FTMP(IKP)
      FTMP(IKP)=F
      GO TO 113
  117 IF(ABS(RHO-X2).LT..01*TAU)GO TO 114
      IF(RHO.LT.X2)GO TO 118
      X3=RHO
      FTMP(IKP+1)=F
      GO TO 113
  118 X1=RHO
      FTMP(IKP-1)=F
      GO TO 113
  114 DO 115 I=1,NVAR
      Y(I)=Y(I)+X2*Z(I)
  115 VAL(I)=Y(I)
      CALL GRAD(NPL,NAL,NN,JI,KI,JO,KO,MP,ELT,VAL,LIN,NOM,OMGMIN,NVAR,
     1OMGMAX,D,G,FUNC,1,2,WM)
      IF((FKP-FUNC).LT.EPS4)GO TO 500
      FKP=FUNC
      KOUT=KOUT+1
      IF(KOUT.GE.KWIT)GO TO 499
      GO TO 260
  107 DO 108 I=1,NVAR
  108 Y(I)=VAL(I)
```

```
      CALL GRAD(NPL,NAL,NN,JI,KI,JO,KC,MP,ELT,VAL,LIN,NOM,OMGMIN,NVAR,
     1OMGMAX,D,G,FUNC,1,2,WM)
      IF((FKP-FUNC).LT.EPS4)GO TO 500
      FKP=FUNC
      KOUT=KOUT+1
      IF(KOUT.GE.KWIT)GO TO 499
      GO TO 260
C-----INTERPOLATION HAS CONVERGED, CHECK FOR CONVERGENCE, IF NOT, SEE
C     IF NQ SHOULD BE DROPPED FROM THE BASIS
  106 DO 111I=1,NVAR
      Y(I)=Y(I)+TAU*Z(I)
  111 VAL(I)=Y(I)
      CALL GRAD(NPL,NAL,NN,JI,KI,JO,KC,MP,ELT,VAL,LIN,NOM,OMGMIN,NVAR,
     1OMGMAX,D,G,FUNC,1,2,WM)
      IF((FKP-FUNC).LT.EPS4)GO TO 500
      FKP=FUNC
      KOUT=KOUT+1
      IF(KOUT.GE.KWIT)GO TO 499
      IF(JSAME.EQ.0) GO TO 305
      CALL ACTIVE
C-----DETERMINE WHICH CONSTRAINTS ARE IN JCON BUT NOT IN ICON
      IF(NAC.NE.0) GO TO 321
      DO 322 I=1,NMACT
  322 KCON(I) = JCON(I)
      K=NMACT
      GO TO 323
  321 K=0
      DO 320 I=1,NMACT
      DO 319  J=1,NAC
      IF(ICON(J).EQ.JCON(I)) GO TO 320
  319 CONTINUE
      K=K+1
      KCON(K) = JCON(I)
  320 CONTINUE
  323 DO 324 I=1,K
  324 JCON(I) = KCON(I)
      J=0
  330 DO 8 IJK=1,NVAR
    8 VAL(IJK)=Y(IJK)
      CALL GRAD(NPL,NAL,NN,JI,KI,JO,KC,MP,ELT,VAL,LIN,NOM,OMGMIN,NVAR,
     1OMGMAX,D,G,FUNC,1,2,WM)
      CALL FINDZ
      IF(DN.LE.EPS1) GO TO 492
      CALL THETA
      IF(THMIN.GE.0.)GO TO 335
      CALL ADDHY
      J=J+1
      IF(J.EQ.K) GO TO 335
```

```
      GC TO 330
  305 CALL ADDHY
  335 DC 338 I=1,NAC
      JQ = ICON(I)
      DLAM(I) = -B(JQ)
      DO 338 K=1,NVAR
  338 DLAM(I) = DLAM(I) + ENQ(K,I)*Y(K)
C-----CLAM NOW CONTAINS THF LAMBDA VALUES
      CALL CORREC
C-----Y HAS NOW BEEN CORRECTED, PRINT NEW VALUES OF LAMBDAS IF CHECK IS DESIRED
C-----CHECK TO SEE IF A HYPERPLANE SHOULD BE DROPPED,AFTER CHECKING CONV
C-----CALCULATE GAMV, THE NORM OF THE EINV MATRIX, AND RQ
  260 CALL FINDZ
      LCON = 0
      IF(DN.LE.ECON) LCON = 1
C-----DN LE ECON MEANS (4.11) IS SATISFIED; DETERMINE RQ AND MAX. COMPONENT
      IF(NAC.EQ.0) GO TO 328
      DO 302 I=1,NAC
      D2R(I) = 0.
      DO 302 K=1,NVAR
  302 D2R(I) = D2R(I) + ENQ(K,I)* G(K)
      DO 303 I=1,NAC
      D1R(I) = 0.
      DO 303 K=1,NAC
  303 D1R(I) = D1R(I) + EINV(I,K)*D2R(K)
      RMAX = -1.E3
      DO 261 I=1,NAC
      IF(D1R(I).LE.RMAX) GO TO 261
      RMAX = D1R(I)
      IDROP = I
  261 CONTINUE
      IF(RMAX.GT.0.) GO TO 270
      IF(LCON.EQ.1) GO TO 500
C-----LCON =1 AND RMAX LESS THAN 0. MEANS PROCESS HAS CONVERGED
  270 LCON = 0
      ROMAX = 0.
      TRACE = 0.
      DO 262 I=1,NAC
      TRACE = TRACE + EINV(I,I)
      ROW = 0.
      DO 263 J=1,NAC
  263 ROW = ROW +  ABS(EINV(I,J))
      IF(ROW.GT.ROMAX) ROMAX = ROW
  262 CONTINUE
      IF(TRACE.LT.ROMAX) ROMAX = TRACE
      EDROP = .5*RMAX/( SQRT(ROMAX))
      IF(DN.GT.EDROP) GO TO 180
C-----EDROP DETERMINED FROM EQ.(4.7)
```

```
C-----DROP HYPERPLANE IDROP FROM THE BASIS, RE-FORM ENQ, EINV
      CALL DROP
      CALL FINDZ
      GO TO 180
  328 IF(LCON.EQ.1) GO TO 550
      GO TO 180
  490 WRITE(6,1003)
      GO TO 550
  491 WRITE(6,1004)
      GO TO 550
  492 WRITE(6,1000)
      GO TO 550
  550 CONTINUE
      GO TO 500
  499 WRITE(6,127)KOUT
  127 FORMAT(1HO' TERMINATED UPON MEETING MAXIMUM NO. OF EVALUATIONS. KW
     1IT =',I5)
  500 DO 9 IJK=1,NVAR
    9 VAL(IJK)=Y(IJK)
      WRITE(6,1005)(VAL(I),I=1,NPL)
      TTT=0.
      DO 119 I=1,NVAR
  119 TTT=G(I)**2+TTT
      TNORM=SQRT(TTT)
      WRITE(6,1006)FUNC,TNORM,KOUT
      CALL GRAD(NPL,NAL,NN,JI,KI,JO,KC,MP,ELT,VAL,LIN,NOM,OMGMIN,NVAR,
     1OMGMAX,D,G,COST,2,2,WM)
 1005 FORMAT(5X' Y = ',8(E15.7)/)
      STOP
      END


      SUBROUTINE ADDHY
C-----ADDS A HYPERPLANE TO THE GROUP OF ACTIVE CONSTRAINTS,CALCULATES
C-----THE NQT*NQ INVERSE MATRIX, ADDS NQ+1 TO THE BASIS AND CALCULATES
C-----PQ. THE NO. OF CONSTRAINING HYPERPLANES INITIALLY IN THE BASIS IS
C-----NAC. THE INDEX OF THE BASIS VECTOR TO BE ADDED IS IMAX.IDEP = 1
C-----INDICATES THAT THE HYPERPLANE IS LINEARLY DEPENDENT AND THUS WAS
C-----NOT ADDED.
      COMMON EHQ(15,32),ENQ(15,15),PQ(15,15),EINV(15,15),
     1D1W(15,15),B(32),D1R(65),D2R(65),Y(30),Z(65),VNQ(30),
     2YTMP(15),GTMP(15),YTRY(15),GTRY(15),DLAM(32),G(15),
     4EPS1,EPS2,EPS3,AZ,DB4,THMIN,HMIN,DN,DZ,DM,DS,DT,RHO,DY,RP,RE,
     5ICON(32),KCON(32),JCON(32),MCON(32),ISAME(30),NVAR,NCON,
     6IDROP,IMAX,NAC,IDEP,NACM1,JV,NMACT,NQP1,NS,NSP1,NVAR2,IND
      DO 228 K=1,NVAR
  228 VNQ(K) = EHQ(K,IMAX)
      IDEP = 0
```

```
      IF(NAC.NE.0) GO TO 52
C-----FORM P1 AND EINV 1
      DO 53 I=1,NVAR
      DO 53 J=1,NVAR
      PQ(I,J) = -VNQ(I)*VNQ(J)
      IF(I.EQ.J) PQ(I,J) = PQ(I,J) + 1.
   53 CONTINUE
      EINV(1,1) = 1.
      L    = NAC + 1
      GO TO 54
   52 DO 1 I=1,NAC
    1 D2R(I) = 0.
      DO 2 I=1,NAC
      DO 2 K=1,NVAR
    2 D2R(I) = D2R(I) + ENQ(K,I)*VNQ(K)
      DO 3 I=1,NAC
    3 D1R(I)=0.
      DO 4 I=1,NAC
      DO 4 K=1,NAC
    4 D1R(I) = D1R(I) + EINV(I,K)* D2R(K)
C-----RQ-1 IN D1R
      DO 5 I=1,NVAR
    5 D2R(I) = 0.
      DO 6 I=1,NVAR
      DO 6 J=1,NVAR
    6 D2R(I) = D2R(I) + PQ(I,J) * VNQ(J)
      AZ = 0.
      DO 7 I=1,NVAR
    7 AZ = AZ + D2R(I)* D2R(I)
      IF(AZ.LE.EPS1) GO TO 50
C-----EPS1 IS THRESHHOLD FOR LINEAR DEPENDENCE
      DO 9 J=1,NAC
      DO 9 I=1,NAC
    9 D1W(I,J) = D1R(I)*D1R(J)/AZ + EINV(I,J)
C-----D1W CONTAINS THE MATRIX B1 OF EQ.(3.11)
      DB4 = 1./AZ
      DO 11 JK=1,NAC
   11 D1R(JK) = -DB4*D1R(JK)
C-----B2 IS IN D1R,AND B3 IS B2 TRANSPOSED
      DO 12  JK=1,NAC
      DO 12  KK=1,NAC
   12 EINV(JK,KK) = D1W(JK,KK)
      L = NAC+1
      DO 13 JK=1,NAC
      EINV(JK,L) = D1R(JK)
   13 EINV(L,JK) = D1R(JK)
      EINV(L,L)  = DB4
C-----THIS COMPLETES THE DETERMINATION OF NQT*NQ INVERSE, ADD NQ TO
```

```
C-----BASIS AND CALCULATE PQ USING EQ.(3.17)
      DO 15  JK=1,NVAR
      DO 15  KK=1,NVAR
   15 PQ(JK,KK) =-D2R(JK)*D2R(KK)/AZ + PQ(JK,KK)
C-----PQ HAS NOW BEEN OBTAINED FROM PQ-1
   54 NAC = NAC+1
      ICON(NAC) = IMAX
      DO 14 JK=1,NVAR
   14 ENQ(JK,L) = VNQ(JK)
      GO TO 51
   50 IDEP = 1
   51 CONTINUE
   60 CONTINUE
      RETURN
      END

      SUBROUTINE DROP
C-----DROPS HYPERPLANE IDROP FROM THE GROUP OF ACTIVE CONSTRAINTS
C-----CALCULATES THE NEW NQT*NQ INVERSE AND THE NEW PQ MATRICES AND
C-----DROPS NL FROM ENQ
      COMMON ENQ(15,32),ENQ(15,15),PQ(15,15),EINV(15,15),
     1D1W(15,15),B(32),D1R(65),D2R(65),Y(30),Z(65),VNQ(30),
     2YTMP(15),GTMP(15),YTRY(15),GTRY(15),DLAM(32),G(15),
     4EPS1,EPS2,EPS3,AZ,DB4,THMIN,HMIN,DN,DZ,DM,DS,DT,RHO,DY,RP,RE,
     5ICON(32),KCON(32),JCON(32),MCON(32),ISAME(30),NVAR,NCON,
     6IDROP,IMAX,NAC,IDEP,NACM1,JV,NMACT,NQP1,NS,NSP1,NVAR2,IND
      IF(NAC.NE.1) GO TO 20
      DO 21 I=1,NVAR
      DO 21 J=1,NVAR
      PQ(I,J) = C.
      IF(I.EQ.J) PQ(I,J) = 1.
   21 CONTINUE
      NAC = C
      GO TO 15
C-----DROP NL FROM BASIS
   20 L=IDROP
C-----INTERCHANGE LTH ROW AND QTH ROW OF EINV AND THE LTH COL AND QTH
C-----COL OF EINV.
      IF(L.EQ.NAC) GO TO 16
    3 DO 4 I=1,NAC
      DN = EINV(NAC,I)
      EINV(NAC,I) =EINV(L,I)
    4 EINV(L,I) = DN
      DO 5 I=1,NAC
      DN = EINV(I,NAC)
      EINV(I,NAC) = EINV(I,L)
    5 EINV(I,L) = DN
```

```
C-----NOW READY TO DROP NL
   16 NACM1 = NAC-1
      DO 6 IK=1,NACM1
      DO 6 JK=1,NACM1
    6 D1W(IK,JK) = EINV(IK,JK)
      DO 7 I=1,NACM1
    7 D2R(I) = EINV(I,NAC)
      DB4 = EINV(NAC,NAC)
C-----B4 IS IN DB4,B1 IN D1W, AND B2 IN D2R
      DO 8 JK=1,NACM1
      DO 8 KK=1,NACM1
    8 EINV(JK,KK) = D1W(JK,KK) - D2R(JK)*D2R(KK)/DB4
C-----THE NEW INVERSE IS NOW IN EINV, DROP NL FROM ENQ
      IF(L.EQ.NAC) GO TO 12
      ICON(L) = ICON(NAC)
      DO 9 J=1,NVAR
    9 ENQ(J,L) = ENQ(J,NAC)
   12 NAC = NAC-1
C-----DETERMINE NEW PROJECTION MATRIX PQ
      DO 10 I=1,NAC
      DO 10 J=1,NVAR
      D1W(I,J) = 0.
      DO 10 K=1,NAC
   10 D1W(I,J) = D1W(I,J) + EINV(I,K)* ENQ(J,K)
      DO 11 I=1,NVAR
      DO 11 J=1,NVAR
      PQ(I,J) = 0.
      IF(I.EQ.J) PQ(I,J)=1.
      DO 11 K=1,NAC
   11 PQ(I,J) = PQ(I,J) - ENQ(I,K)* D1W(K,J)
   15 CONTINUE
C-----NEW PQ MATRIX COMPUTED
      RETURN
      END


      SUBROUTINE VIOLA
C-----DETERMINES WHICH CONSTRAINTS ARE VIOLATED
      COMMON EHQ(15,32),ENQ(15,15),PQ(15,15),EINV(15,15),
     1D1W(15,15),B(32),D1R(65),D2R(65),Y(30),Z(65),VNQ(30),
     2YTMP(15),GTMP(15),YTRY(15),GTRY(15),DLAM(32),G(15),
     4EPS1,EPS2,EPS3,AZ,DB4,THMIN,HMIN,DN,DZ,DM,DS,DT,RHC,DY,RP,RE,
     5ICON(32),KCON(32),JCON(32),MCON(32),ISAME(30),NVAR,NCON,
     6IDROP,IMAX,NAC,IDEP,NACM1,JV,NMACT,NQP1,NS,NSP1,NVAR2,IND
      DO 223 J=1,NCON
      D1R(J) = 0.
      DO 224 I=1,NVAR
  224 D1R(J) = D1R(J) + EHQ(I,J)* Y(I)
```

```
  223 D1R(J) = D1R(J) - B(J)
C-----CHECK SIGN OF D1R
      JV=0
      DO 225 I=1,NCON
      IF(D1R(I).GE.-EPS1) GO TO 225
      JV=JV+1
      KCON(JV) = I
      DLAM(JV) = D1R(I)
  225 CONTINUE
      RETURN
      END

      SUBROUTINE ACTIVE
      COMMON EHQ(15,32),ENQ(15,15),PQ(15,15),EINV(15,15),
     1D1W(15,15),B(32),D1R(65),D2R(65),Y(30),Z(65),VNQ(30),
     2YTMP(15),GTMP(15),YTRY(15),GTRY(15),DLAM(32),G(15),
     4EPS1,EPS2,EPS3,AZ,DB4,THMIN,HMIN,DN,DZ,DM,DS,DT,RHO,DY,RP,RE,
     5ICON(32),KCON(32),JCON(32),MCON(32),ISAME(30),NVAR,NCON,
     6IDROP,IMAX,NAC,IDEP,NACM1,JV,NMACT,NQP1,NS,NSP1,NVAR2,IND
      K=0
      DO 152 J=1,NCON
      D1R(J) = -B(J)
      DO 153 M=1,NVAR
  153 D1R(J) = D1R(J) +EHQ(M,J)*Y(M)
C-----CHECK SIGN OF D1R
      IF(D1R(J).GE.1.D-10) GO TO 152
      K=K+1
      JCON(K) = J
  152 CONTINUE
      NMACT=K
      RETURN
      END

      SUBROUTINE FINDZ
      COMMON EHQ(15,32),ENQ(15,15),PQ(15,15),EINV(15,15),
     1D1W(15,15),B(32),D1R(65),D2R(65),Y(30),Z(65),VNQ(30),
     2YTMP(15),GTMP(15),YTRY(15),GTRY(15),DLAM(32),G(15),
     4EPS1,EPS2,EPS3,AZ,DB4,THMIN,HMIN,DN,DZ,DM,DS,DT,RHO,DY,RP,RE,
     5ICON(32),KCON(32),JCON(32),MCON(32),ISAME(30),NVAR,NCON,
     6IDROP,IMAX,NAC,IDEP,NACM1,JV,NMACT,NQP1,NS,NSP1,NVAR2,IND
      DN = 0.
      DO 154 I=1,NVAR
  154 D1R(I) = 0.
      DO 155 J=1,NVAR
      DO 155 K=1,NVAR
  155 D1R(J)=D1R(J) + PQ(J,K)*G(K)
```

```
      DO 157 I=1,NVAR
  157 DN = DN + D1R(I)**2
      DN =  SQRT(DN)
      IF( ABS(DN).LE.1.D-10)  GO TO 200
      DO 156 I=1,NVAR
  156 Z(I) = D1R(I)/DN
      GO TO 300
  200 DO 201 I=1,NVAR
  201 Z(I) = 0.
  300 CONTINUE
      RETURN
      END

      SUBROUTINE THETA
      COMMON EHQ(15,32),ENQ(15,15),PQ(15,15),EINV(15,15),
     1D1W(15,15),B(32),D1R(65),D2R(65),Y(30),Z(65),VNQ(30),
     2YTMP(15),GTMP(15),YTRY(15),GTRY(15),DLAM(32),G(15),
     4EPS1,EPS2,EPS3,AZ,DB4,THMIN,HMIN,DN,DZ,DM,DS,DT,RHO,DY,RP,RE,
     5ICON(32),KCON(32),JCON(32),MCON(32),ISAME(30),NVAR,NCON,
     6IDROP,IMAX,NAC,IDEP,NACM1,JV,NMACT,NQP1,NS,NSP1,NVAR2,IND
      THMIN = 1.E6
      IF(NMACT.EQ.0) GO TO 100
      DO 157 I=1,NMACT
      DN = 0.
      JQ = JCON(I)
      IF(NAC.EQ.0) GO TO 160
      DO 159 J=1,NAC
      IF(JQ.EQ.ICON(J)) GO TO 157
  159 CONTINUE
  160 DO 158 K=1,NVAR
  158 DN = DN + Z(K)*EHQ(K,JQ)
      IF(DN.GE.THMIN) GO TO 157
      THMIN = DN
      IMAX = JQ
  157 CONTINUE
  100 CONTINUE
      RETURN
      END

      SUBROUTINE CORREC
C-----FORM NQ(NQT*NQ) INVERSE FOR USE IN EQ.(7.4)
      COMMON EHQ(15,32),ENQ(15,15),PQ(15,15),EINV(15,15),
     1D1W(15,15),B(32),D1R(65),D2R(65),Y(30),Z(65),VNQ(30),
     2YTMP(15),GTMP(15),YTRY(15),GTRY(15),DLAM(32),G(15),
     4EPS1,EPS2,EPS3,AZ,DB4,THMIN,HMIN,DN,DZ,DM,DS,DT,RHO,DY,RP,RE,
     5ICON(32),KCON(32),JCON(32),MCON(32),ISAME(30),NVAR,NCON,
```

```
     6IDROP,IMAX,NAC,IDEP,NACM1,JV,NMACT,NQP1,NS,NSP1,NVAR2,IND
      IF(NAC.EQ.0) GO TO 100
      DO 229 I=1,NVAR
      DO 229 J=1,NAC
      D1W(I,J) = 0.
      DO 229 K=1,NAC
  229 D1W(I,J) = D1W(I,J) + ENQ(I,K)* EINV(K,J)
      DO 231 I=1,NVAR
      DO 231 K=1,NAC
  231 Y(I) = Y(I) - D1W(I,K)*DLAM(K)
  100 CONTINUE
      RETURN
      END
C                              SECTION (2)
C
C     THE MODIFIED CALAHAN PROGRAM FOR PRODUCING THE ACTUAL FREQUENCY
C     RESPONSE
C

      SUBROUTINE ASBS(LN,C,G,H,X,NP,MG,I,KEY1,JN,JP,II,NPL,NAL)
      DIMENSION C(50),G(50),H(50),X(60),Z(60),MG(30,5),Y(3,50)
      IF(I-2)101,100,101
100   GO TO(2,3,3,4),KEY1
    3 NP=2*LN-1
      JM=1
      JP=2
      GO TO 5
    4 NP=2*LN-3
      JM=2
      JP=3
      GO TO 5
    2 NP=2*LN+1
      JM=0
      JP=1
5     JN=LN-JM
  101 N=MG(JP,3)
      IF(NP-2)30,30,102
30    IF(NP)31,31,32
32    X(1)=0.
      NP=1
      RETURN
31    X(1)=0.
      NP=1
      RETURN
102   DO 130 J=1,NP
130   X(J)=0.
      Z(1)=1.
```

```
      K1=1
      K2=1
      DO 1 J=1,JN
      K=JM+J
      N=MG(K,3)
      K2P=K2+2
      DO 116 K=1,3
      DO 117 I=K1,K2P
117   Y(K,I)=0.
116   CONTINUE
      IF(C(N))81,80,81
81    N1=2
      N2=2
      DO 110 K=K1,K2
110   Y(3,K+2)=Z(K)*C(N)
      IF(G(N))86,87,86
80    IF(G(N))82,83,82
82    N2=1
86    N1=1
      DO 111 K=K1,K2
111   Y(2,K+1)=Z(K)*G(N)
87    IF(H(N))85,84,85
83    IF(H(N))88,89,88
88    N2=0
85    N1=0
      DO 112 K=K1,K2
112   Y(1,K)=Z(K)*H(N)
84    K1=K1+N1
      K2=K2+N2
      DO 113 K=K1,K2
113   Z(K)=Y(1,K)+Y(2,K)+Y(3,K)
1     CONTINUE
      IF(II)132,133,133
132   DO 131 J=K1,K2
131   X(J)=-Z(J)
      RETURN
133   DO 134 J=K1,K2
134   X(J)=Z(J)
89    RETURN
      END


      SUBROUTINE UTPLOT (X   ,Y   ,NDATA,RANGE,KKZ)
      DIMENSION GRID(61,101),XSCALE(6),YSCALE(7)
      DIMENSION X   (1),Y   (1),RANGE(4)
      INTEGER*2  ZERO/1H0/,IYORG(101)
      INTEGER*2  GRID,BLANK,XCHAR,DOT
      DATA  DOT,XCHAR,BLANK  /2H. ,2HX ,2H  /
```

```
C
CC      GRID IS THE MATRIX USED TO PLOT THE POINTS
C
        DO 8   I=1,101
      8 IYORG(I) = BLANK
        IERR=0
        XMAX=RANGE(1)
        XMIN=RANGE(2)
        YMAX=RANGE(3)
        YMIN=RANGE(4)
C
CC      CHECKING X AND Y POINTS AND PLOTTING THOSE OUT OF RANGE
CC      AT THE MARGIN
C
        DO 30 I=1,NDATA,KKZ
        IF(X  (I)-XMAX) 205,205,220
   220  X  (I)=XMAX
        IERR=IERR+1
        GOTO 210
   205  IF(X  (I)-XMIN)203,210,210
   203  X  (I)=XMIN
        IERR=IERR+1
   210  IF(Y  (I)-YMAX)215,215,212
   212  Y  (I)=YMAX
        IERR=IERR+1
        GOTO 30
   215  IF(Y  (I)-YMIN)217,  30,30
   217  Y  (I)=YMIN
        IERR=IERR+1
C
    30  CONTINUE
C
CC      PLOTTING X AND Y AXIS , IF NECESSARY
C
        XRANGE=XMAX-XMIN
        YRANGE=YMAX-YMIN
C
CC      BLANKING OUT MATRIX-(GRID)
C
        DO 300 I=1,61
        DO 301 JJ=1,101
  301   GRID(I,JJ)=BLANK
   300  CONTINUE
        YTEST=YMAX*YMIN
        XTEST=XMAX*XMIN
        IF(XTEST)1,222,222
   222  IF(YTEST)333,444,444
     1  IYAXIS = 100.*(-XMIN)/XRANGE+1.5
```

```
      IYORG(IYAXIS-1) = ZERO
      IYORG(IYAXIS) = DOT
      IYORG(IYAXIS+1) = ZERO
      DO 40 I=1,61
   40 GRID(I,IYAXIS)=DOT
      GO TO 222
  333 IXAXIS=60.*YMAX/YRANGE+1.5
      DO 60 I=1,101
   60 GRID(IXAXIS,I)=DOT
C
CC    PLACING POINTS IN THEIR PROPER GRID POSITIONS
C
  444 DO 70 I=1,NDATA,KKZ
      IPTX=60.*(YMAX-Y(I))/YRANGE+1.5
      IPTY = 100.*(X(I)-XMIN)/XRANGE+1.5
   70 GRID(IPTX,IPTY)=XCHAR
C
CC    COMPUTE PROPER SCALE NUMBERS
C
 8000 XINCR=XRANGE/5.
      YINCR=YRANGE/6.
      XSCALE(1)=XMAX
      YSCALE(1)=YMAX
      DO 80 I=2,6
   80 XSCALE(I)=XSCALE(I-1)-XINCR
      DO 81 I=2,7
   81 YSCALE(I)=YSCALE(I-1)-YINCR
C
CC    OUTPUT SECTION WITH GRAPH
C
      WRITE (6,401)  IYORG
      WRITE (6,17) XSCALE(6),XSCALE(5),XSCALE(4),XSCALE(3),XSCALE(2),
     1     XSCALE(1)
   17 FORMAT(12X,1PE10.3,5(10X,E10.3)/15X,2H**,10(10H**********),3H+**)
      II=1
      I=0
      DO 101 IK=1,61
      IF(I)91,91,92
   91 WRITE (6,18) YSCALE(II),(GRID(IK,IX),IX=1,101),YSCALE(II)
   18 FORMAT(3X,1PE10.3,4H  + ,101A1,4H +  ,E10.3)
      II=II+1
      GO TO 102
92    IF (IK .NE. IXAXIS)  GO TO 192
      WRITE (6,400)  (GRID(IK,IX),IX=1,101)
400   FORMAT (8X,4H0.00,3X,1H*,1X,101A1,2H *,3X,4H0.00)
      GO TO 102
  192 WRITE (6,19) (GRID(IK,IX),IX=1,101)
   19 FORMAT(15X,1H*,1X,101A1,1X,1H*)
```

```
  102 I=I+1
      IF(I-10)1C1,103,103
  103 I=0
  101 CONTINUE
      WRITE (6,22) XSCALE(6),XSCALE(5),XSCALF(4),XSCALE(3),XSCALE(2),
     1     XSCALE(1)
   22 FORMAT (15X,2H**, 10(10H+*********),3H+**/, 1P
     1  12X,E10.3, 5(10X,E10.3))
      WRITE (6,4C1)  IYORG
401   FORMAT(17X,1C1A1)
      IF(IERR) 1000,1000,1001
 1001 WRITE (6,20) IERR
  20  FORMAT(10X 'NUMBER OF POINTS OUT OF RANGE ='  I4)
 1000 RETURN
      END
C

      SUBROUTINE DPOLRT (XCCF,COF,M,RCCTR,ROOTI,IER)
C
C        PURPOSE
C           COMPUTES THE REAL AND COMPLEX ROOTS OF A REAL POLYNOMIAL
C
C        USAGE
C           CALL PCLRT(XCOF,COF,M,ROCTR,ROOTI,IER)
C
C        DESCRIPTION OF PARAMETERS
C           XCOF -VECTOR OF M+1 COEFFICIENTS OF THE POLYNOMIAL
C                 CRDERED FROM SMALLEST TO LARGEST POWER
C           COF  -WORKING VECTOR OF LENGTH M+1
C           M    -CRDER OF POLYNOMIAL
C           ROOTR-RESULTANT VECTCR OF LENGTH M CONTAINING REAL ROOTS
C                 OF THE POLYNOMIAL
C           ROOTI-RESULTANT VECTOR OF LENGTH M CONTAINING THE
C                 CORRESPONDING IMAGINARY ROOTS OF THE POLYNOMIAL
C           IER  -ERROR CODE WHERE
C                 IER=0  NO ERROR
C                 IER=1  M LESS THAN CNE
C                 IER=2  M GREATER THAN 36
C                 IER=3  UNABLE TO DETERMINE ROOT WITH 500 INTERATIONS
C                        ON 8 STARTING VALUES
C                 IER=4  HIGH CRDER COEFFICIENT IS ZERC
C
C        REMARKS
C           LIMITED TO 36TH ORDER PCLYNOMIAL OR LESS.
C           FLOATING PCINT OVERFLOW MAY OCCUR FOR HIGH ORDER
C           POLYNOMIALS BUT WILL NOT AFFECT THE ACCURACY OF THE RESULTS.
C
```

```
C            NONE
CC
CC          METHOD
CC             NEWTON-RAPHSON ITERATIVE TECHNIQUE.  THE FINAL ITERATIONS
CC             ON EACH ROOT ARE PERFORMED USING THE ORIGINAL POLYNOMIAL
CC             RATHER THAN THE REDUCED POLYNOMIAL TO AVOID ACCUMULATED
CC             ERRORS IN THE REDUCED POLYNOMIAL.
CCC
C       ..................................................................
C
      DIMENSION XCOF(1),COF(1),ROOTR(1),ROOTI(1)
C
C          ..............................................................
C
C          IF A DOUBLE PRECISION VERSION OF THIS ROUTINE IS DESIRED, THE
C          C IN COLUMN 1 SHOULD BE REMOVED FROM THE DOUBLE PRECISION
C          STATEMENT WHICH FOLLOWS.
C
      DOUBLE PRECISION XCOF,COF,ROOTR,ROOTI,XO,YO,X,Y,XPR,YPR,UX,UY,V,
     1                 YT,XT,U,XT2,YT2,SUMSQ,DX,DY,TEMP,ALPHA
C
CC         THE C MUST ALSO BE REMOVED FROM DOUBLE PRECISION STATEMENTS
CC         APPEARING IN OTHER ROUTINES USED IN CONJUNCTION WITH THIS
CC         ROUTINE.
CC
CC         THE DOUBLE PRECISION VERSION OF THIS SUBROUTINE MUST ALSO
CC         CONTAIN DOUBLE PRECISION FORTRAN FUNCTIONS.  ABS IN STATEMENTS
CC         78 AND 122 MUST BE CHANGED TC DABS.
CCC
C          ..............................................................
C
      IFIT=C
      N=M
      IER=0
      IF(XCOF(N+1))10,25,10
   10 IF(N) 15,15,32
C
CC         SET ERROR CODE TO 1
C
   15 IER=1
   20 RETURN
C
CC         SET ERROR CODE TO 4
C
   25 IER=4
      GO TO 20
C
CC         SET ERROR CODE TO 2
```

```
C
   30 IER=2
      GO TO 20
   32 IF(N-36) 35,35,30
   35 NX=N
      NXX=N+1
      N2=1
      KJ1 = N+1
      DO 40 L=1,KJ1
      MT=KJ1-L+1
   40 COF(MT)=XCOF(L)
C
CC         SET INITIAL VALUES
C
   45 XO=.005001C1
      YO=C.01000101
C
CC         ZERO INITIAL VALUE COUNTER
C
      IN=0
   50 X=XO
C
CC         INCREMENT INITIAL VALUES AND COUNTER
C
      XO=-10.0*YO
      YO=-10.0*X
C
CC         SET X AND Y TC CURRENT VALUE
C
      X=XO
      Y=YO
      IN=IN+1
      GO TO 59
   55 IFIT=1
      XPR=X
      YPR=Y
C
CC         EVALLATE POLYNOMIAL AND DERIVATIVES
C
   59 ICT=0
   60 UX=0.C
      UY=0.0
      V =0.C
      YT=0.C
      XT=1.C
      U=COF(N+1)
      IF(U) 65,130,65
   65 DO 70 I=1,N
```

```
      L =N-I+1
      XT2=X*XT-Y*YT
      YT2=X*YT+Y*XT
      U=U+COF(L )*XT2
      V=V+COF(L )*YT2
      FI=I
      UX=UX+FI*XT*COF(L )
      UY=UY-FI*YT*COF(L )
      XT=XT2
   70 YT=YT2
      SUMSQ=UX*UX+UY*UY
      IF(SUMSQ) 75,110,75
   75 DX=(V*UY-U*UX)/SUMSQ
      X=X+DX
      DY=-(U*UY+V*UX)/SUMSQ
      Y=Y+DY
   78 IF(DABS(DY)+DABS(DX)-1.0E-05) 100,80,80
C
C        STEP ITERATION COUNTER
C
   80 ICT=ICT+1
      IF(ICT-500) 60,85,85
   85 IF(IFIT)100,90,100
   90 IF(IN-5) 50,95,95
C
C        SET ERROR CODE TO 3
C
   95 IER=3
      GO TO 20
  100 DO 105 L=1,NXX
      MT=KJ1-L+1
      TEMP=XCOF(MT)
      XCOF(MT)=COF(L)
  105 COF(L)=TEMP
      ITEMP=N
      N=NX
      NX=ITEMP
      IF(IFIT) 120,55,120
  110 IF(IFIT) 115,50,115
  115 X=XPR
      Y=YPR
  120 IFIT=0
  122 IF(DABS(Y/X)-1.0E-04) 135,125,125
  125 ALPHA=X+X
      SUMSQ=X*X+Y*Y
      N=N-2
      GO TO 140
  130 X=0.0
```

```
      NX=NX-1
      NXX=NXX-1
  135 Y=C.0
      SUMSQ=C.0
      ALPHA=X
      N=N-1
  140 COF(2)=COF(2)+ALPHA*COF(1)
  145 DO 150 L=2,N
  150 COF(L+1)=COF(L+1)+ALPHA*COF(L)-SUMSQ*COF(L-1)
  155 ROOTI(N2)=Y
      ROOTR(N2)=X
      N2=N2+1
      IF(SUMSQ) 160,165,160
  160 Y=-Y
      SUMSQ=0.0
      GO TO 155
  165 IF(N) 20,20,45
      END


      SUBROUTINE FREQ (NA,NB,A,B,G,W)
      DIMENSION A(60),AE(30),AO(30),AE1(30),AO1(30),B(60)
     1,BE(30),BO(30),BE1(30),BO1(30),G(4)
      CALL PARTS (NA,A,M1,AE,N1,AO,M11,AE1,N11,AO1)
      CALL PARTS (NB,B,M2,BE,N2,BO,M12,BE1,N12,BO1)
   33 EVN=SUM (M1,AE,W)
      ODDN=W*SUM (N1,AO,W)
      EV1N=W*SUM (M11,AE1,W)
      ODD1N=SUM (N11,AO1,W)
      EVD=SUM (M2,BE,W)
      ODDD=W*SUM (N2,BO,W)
      EV1D=W*SUM (M12,BE1,W)
      ODD1D=SUM (N12,BO1,W)
      TOP=EVN*EVN+ODDN*ODDN
      BOTTOM=EVD*EVD+ODDD*ODDD
      Y=ODDN*EVD-ODDD*EVN
      X=EVN*EVD+ODDD*ODDN
      IF (ABS(X+Y+TOP+BOTTOM) - 1.0E+65)  1,1,2
    2 FREQQ = W / 6.28318
      WRITE (6,3) W, FREQQ
    3 FORMAT (//,5X,'DUE TO OVERFLOW, THE FREQUENCY RESPONSE DATA ',/,
     19X,'WILL BE UNRELIABLE FOR',/,20X,'W = ',1PE15.6,'  RADIANS',
     2    ' = ' 1PE15.6, ' HZ')
    1 IF (X.EQ. C.C)  X=1.0E-50
      THETA=57.2957795*ATAN (Y/X)
      IF(X)  111,111,112
  111 THETA=THETA-180.
  112 AMPL=TOP/BOTTOM
```

```
 60   DELAY=(EVD*ODD1D+ODDD*EV1D)/BOTTOM-(EVN*ODD1N+ODDN*EV1N)/TOP
      G(1)=W
      G(2)=AMPL
      G(3)=DELAY
      G(4)=THETA
      RETURN
      END

      SUBROUTINE FREQQ(LIN,NOM,OMGMIN,OMGMAX,NA,NB,A,B,KEY1,R2)
      DIMENSION A(60),B(60),G(4),R1(100),R2(100),R3(100),R4(100),R(60)
      MAXPTS = 100
      KL=0
      KC=0
      SL=NOM
      RL=0.0
1     W=RL*(OMGMAX-OMGMIN)/SL+OMGMIN
12    WW=6.28318*W
      CALL FREQ (NA,NB,A,B,G,WW)
      KL=KL+1
      R1(KL) =W
      R2(KL) =4.3429448*ALOG(G(2))
22    R3(KL) =G(4)
      R4(KL) =G(3)
      RL=RL+1.0
      IF (KL-MAXPTS) 86,87,87
87    IF(KC)91,89,189
86    IF(OMGMAX-W)87,87,1
   89 GO TO(93,13),KEY1
  189 GO TO 113
113   WRITE (6,300)
300   FORMAT (1H1,5X, 'CONTINUATION OF ABOVE TABLE AND GRAPH')
      WRITE (6,301)
301   FORMAT(//,2(9X,10HFREQ (HZ)  ,4X,9HGAIN (DB) ,3X, 10HPHASE(DEG) ,
     12X,11HDELAY (SEC) ,2X)//)
      GO TO 91
   13 WRITE(6,8)
8     FORMAT (1H1,2(9X,10HFREQ (HZ) ,4X,9HGAIN (DB) ,3X,10HPHASE(DEG) ,
     12X,11HDELAY (SEC) ,2X)//)
      GO TO 91
128   WRITE (6,300)
      WRITE (6,302)
302   FORMAT(//,2(9X,10HFREQ (HZ)  ,4X,9HOHMS       ,3X,10HPHASE(DEG))//)
      GO TO 91
114   WRITE (6,300)
      WRITE (6,303)
303   FORMAT(//,2(9X,10HFREQ (HZ)  ,4X,9HMHOS       ,3X,10HPHASE(DEG))//)
   91 GO TO 90
```

```
   27 WRITE(6,33) (R1(J),R2(J),R3(J),J=1,KL)
      IF (LIN.EQ.1) CALL PLOT(R1,R2,KL)
      WRITE(6,200)
  200 FORMAT(//,10X,'PLOT OF ABOVE TABLE:  Y-AXIS  OHMS',/,32X,
     1    'X-AXIS  FREQ (HZ) ')
      IF (LIN.EQ.1) CALL PLOT(R1,R3,KL)
      WRITE (6,202)
  202 FORMAT(//,10X,'PLOT OF ABOVE TABLE:  Y-AXIS  PHASE (DEG)',
     1    /,32X,'X-AXIS  FREQ (HZ)')
33    FORMAT(2(F20.7,F15.7,F10.3))
      GO TO 93
   90 WRITE (6,9)  (R1(J),R2(J),R3(J),R4(J),J=1,KL)
9     FORMAT(2(F20.7,F15.7,F10.3,F15.7))
C
      CALL PLOT (R1,R2,KL)
      WRITE (6,203)
      CALL PLOT (R1,R3,KL)
      WRITE (6,204)
      CALL PLOT (R1,R4,KL)
      WRITE (6,205)
      GO TO 93
C
C
  203 FORMAT(//,10X,'PLOT OF ABOVE TABLE:  Y-AXIS  GAIN (DB)',
     1    /,32X,'X-AXIS  FREQ (HZ)')
  204 FORMAT(//,10X,'PLOT OF ABOVE TABLE:  Y-AXIS  PHASE (DEG)',
     1    /,32X,'X-AXIS  FREQ (HZ)')
  205 FORMAT(//,10X,'PLOT OF ABOVE TABLE:  Y-AXIS  DELAY (SEC)',
     1    /,32X,'X-AXIS  FREQ (HZ)')
C
93    IF(OMGMAX-W)94,94,95
95    KL=0
      KC=KC+1
      GO TO 1
94    RETURN
      END
C


      SUBROUTINE GROUP(G,C,H,ML,MP,MAP,ELT,ELTA,VAL,VALA,NPL,NAL,NE)
      DIMENSION MP(100,3),ML(50,5),ELT(100),
     1MAP(20,5),ELTA(20),VAL(100),VALA(20),C(50),G(50),
     2H(50)
      REAL IHC / 4HC   /, IHR/4HR   /, IHG /4HG   /
C     MAKE LIST OF ELEMENTS AND NODE NUMBERS IN BOTH CURRENT AND VOLTAGE
C     GRAPH
      KK=0
      NLL=NPL+NAL
```

```
C      MAKE PARALLEL RLC ELEMENTS INTO SINGLE TOPOLOGICAL ELEMENT
       DO 120 K=1,NLL
       G(K)=0.
       C(K)=0.
  120  H(K)=0.
C      PICK JTH ELEMENT
       DO 109 J=1,NPL
          IF (ELT(J) .EQ. IHC)  GO TO 197
196    IF(VAL(J))197,195,197
195    VAL(J)=.00001
       WRITE (6,194) J
194    FORMAT(//,10X,4HTHE ,I2,50HTH ELEMENT VALUE HAS BEEN REPLACED BY
      1.00001          )
197    ML(J,1)=MP(J,1)
       ML(J,2)=MP(J,2)
       DO 121 K=1,J
C      COMPARE JTH WITH PRECEDING ELEMENTS (BY NODE NUMBER)
       IF(MP(J,1)-ML(K,1))121,103,121
  103  IF(MP(J,2)-ML(K,2))121,104,121
  104  IF(J-K)111,110,111
  110  JK=J-KK
C      IF NOT SAME AS PRECEDING ELEMENTS,PLACE IN PERMANENT LIST OF ELETS
       ML(JK,1)=MP(J,1)
       ML(JK,2)=MP(J,2)
       ML(JK,3)=JK
    2  ML(JK,4)=MP(J,1)
       ML(JK,5)=MP(J,2)
       MP(J,3)=JK
C      TEST FOR TYPE OF ELEMENT AND STORE VALUE
    1  IF(ELT(J) .NE. IHR) GO TO 106
  105  G(JK)=1./VAL(J)+G(JK)
       GO TO 109
  106  IF(ELT(J) .NE. IHC) GO TO 108
  107  C(JK)=VAL(J)+C(JK)
       GO TO 109
  108  H(JK)=1./VAL(J)+H(JK)
       GO TO 109
C      IF SAME AS PRECEDING ELEMENT,ADD VALUE TO THAT OF PRECEDING
111    MP(J,3)=K
       KK=KK+1
          IF (ELT(J) .NE. IHR)  GO TO 116
  115  G(K)=1./VAL(J)+G(K)
       GO TO 109
  116  IF(ELT(J) .NE. IHC) GO TO 118
  117  C(K)=VAL(J)+C(K)
       GO TO 109
  118  H(K)=1./VAL(J)+H(K)
       GO TO 109
```

```
  121 CONTINUE
  109 CONTINUE
      NPE=NPL-KK
      IF(NAL)100,99,100
C     TRANSFER ACTIVE ELEMENT NODE NUMBERS TO PERMANENT LIST
  100 DO 98 J=1,NAL
      NM=NPE+J
      ML(NM,1)=MAP(J,1)
      ML(NM,2)=MAP(J,2)
      ML(NM,3)=NM
      ML(NM,4)=MAP(J,3)
      ML(NM,5)=MAP(J,4)
      MAP(J,5)=NM
         IF (ELT(J) .NE. IHG)  GO TO 11
   10 G(NM)=VALA(J)+G(NM)
      GO TO 98
   11 H(NM)=VALA(J)+H(NM)
   98 CONTINUE
   99 NE=NPL+NAL-KK
      RETURN
      END
C


      FUNCTION IGN(NN,LN,MG)
      DIMENSION MG(30,5),ME(20,2),MM(100)
      K1=1
      K2=2
      K3=1
   19 DO 21 J=1,NN
   21 MM(J)=0
      K=0
      MM(1)=1
    9 KK=K
      DO 7 J=1,LN
      M=MG(J ,K1)
      N=MG(J ,K2)
      IF(MM(N))1,1,3
    1 IF(MM(M))7,7,4
    3 IF(MM(M))14,14,7
    4 ME(N,K3)=-MG(J,3)
      MM(N)=MM(N)+1
   30 K=K+1
      IF(K-LN)7,10,10
   14 ME(M,K3)=MG(J,3)
      MM(M)=MM(M)+1
      GO TO 30
    7 CONTINUE
```

```
      IF(KK-K)9,10,9
   10 IF(K1-4)81,80,81
   81 K1=4
      K2=5
      K3=2
      GO TO 19
80    KZ=2
   35 DO 31 J=2,NN
      DO 32 K=2,NN
      JJJ=J
      KKK=K
      IF(IABS (ME(J,1))-IABS (ME(K,2)))32,33,32
   33 IF(J-K)34,31,34
   32 CONTINUE
   31 CONTINUE
      GO TO 55
34    MY=ME(KKK,2)
      ME(KKK,2)=ME(JJJ,2)
      ME(JJJ,2)=MY
      KZ=KZ+1
      GO TO 35
   55 DO 40 J=2,NN
      IF(ME(J,1)-ME(J,2))41,40,41
   41 KZ=KZ+1
   40 CONTINUE
       IGN=(-1)**KZ
      RETURN
      END


      SUBROUTINE MULLER(ZRO,N8,Z)
C
C        ZRO     COEFF IN ASCENDING ORDER
C        N8    =  ORDER OF POLNOMIAL PLUS ONE
C        Z(I,1)  =  REAL ROOT
C        Z(I,2)  =  CORRESPONDING IMAG PART OF ROOT
C
      DIMENSION ZRO(60),COE(60),Z(60,2)
      N1=N8-1
      IF(N1)1000,1000,1001
C1000 WRITE(6,1119)
1119  FORMAT(//,20X,4HNONE   )
 1000 GO TO 300
1001  DO 1003 J=1,N8
      K=N8-J+1
1003  COE(J)=ZRO(K)
      N2=N1+1
      N4=0
```

```
      I=N1+1
   19 IF(COE(I))9,7,9
    7 N4=N4+1
      Z(N4,1)=0.
      Z(N4,2)=0.
      I=I-1
      IF(N4-N1)19,37,19
    9 CONTINUE
   10 AXR=0.8
      AXI=0.
      L=1
      N3=1
      ALP1R=AXR
      ALP1I=AXI
      M=1
      GOTO99
   11 BET1R=TEMR
      BET1I=TEMI
      AXR=0.85
      ALP2R=AXR
      ALP2I=AXI
      M=2
      GOTO99
   12 BET2R=TEMR
      BET2I=TEMI
      AXR=0.9
      ALP3R=AXR
      ALP3I=AXI
      M=3
      GOTO99
   13 BET3R=TEMR
      BET3I=TEMI
   14 TE1=ALP1R-ALP3R
      TE2=ALP1I-ALP3I
      TE5=ALP3R-ALP2R
      TE6=ALP3I-ALP2I
      TEM=TE5*TE5+TE6*TE6
      IF (TEM .EQ. 0.0)  TEM = 1.0E-35
      TE3=(TE1*TE5+TE2*TE6)/TEM
      TE4=(TE2*TE5-TE1*TE6)/TEM
      TE7=TE3+1.
      TE9=TE3*TE3-TE4*TE4
      TE10=2.*TE3*TE4
      DE15=TE7*BET3R-TE4*BET3I
      DE16=TE7*BET3I+TE4*BET3R
      TE11=TE3*BET2R-TE4*BET2I+BET1R-DE15
      TE12=TE3*BET2I+TE4*BET2R+BET1I-DE16
      TE7=TE9-1.
```

```
      TE1=TE9*BET2R-TE10*BET2I
      TE2=TE9*BET2I+TE10*BET2R
      TE13=TE1-BET1R-TE7*BET3R+TE10*BET3I
      TE14=TE2-BET1I-TE7*BET3I-TE10*BET3R
      TE15=DE15*TE3-DE16*TE4
      TE16=DE15*TE4+DE16*TE3
      TE1=TE13*TE13-TE14*TE14-4.*(TE11*TE15-TE12*TE16)
      TE2=2.*TE13*TE14-4.*(TE12*TE15+TE11*TE16)
      TEM=SQRT (TE1*TE1+TE2*TE2)
      IF(TE1)113,113,112
  113 TE4=SQRT (.5*(TEM-TE1))
      IF(TE4.EQ.0.0)TE4=1.E-35
      TE3=.5*TE2/TE4
      GO TO 111
  112 TE3=SQRT (.5*(TEM+TE1))
      IF(TE2)110,200,200
  110 TE3=-TE3
  200 TE3A = TE3
      IF (TE3A .EQ. 0.0)  TE3A = 1.0E-35
      TE4=.5*TE2/TE3A
  111 TE7=TE13+TE3
      TE8=TE14+TE4
      TE9=TE13-TE3
      TE10=TE14-TE4
      TE1=2.*TE15
      TE2=2.*TE16
      IF(TE7*TE7+TE8*TE8-TE9*TE9-TE10*TE10)204,204,205
  204 TE7=TE9
      TE8=TE10
  205 TEM=TE7*TE7+TE8*TE8
      IF(TEM.EQ.0.0)TEM=1.0E-35
      TE3=(TE1*TE7+TE2*TE8)/TEM
      TE4=(TE2*TE7-TE1*TE8)/TEM
      AXR=ALP3R+TE3*TE5-TE4*TE6
      AXI=ALP3I+TE3*TE6+TE4*TE5
      ALP4R=AXR
      ALP4I=AXI
      M=4
      GO TO 99
   15 N6=1
   38 IF (ABS (HELL)+ABS (BELL)-1.E-20) 18,18,16
   16 TE7=ABS (ALP3R-AXR)+ABS (ALP3I-AXI)
      AXA = AXR
      IF ((AXR .EQ. 0.0) .AND. (AXI .EQ. 0.0))  AXA = 1.0E-35
      IF(TE7/(ABS (AXA)+ABS (AXI))-1.E-7)18,18,17
   17 N3=N3+1
      ALP1R=ALP2R
      ALP1I=ALP2I
```

```
      ALP2R=ALP3R
      ALP2I=ALP3I
      ALP3R=ALP4R
      ALP3I=ALP4I
      BET1R=BET2R
      BET1I=BET2I
      BET2R=BET3R
      BET2I=BET3I
      BET3R=TEMR
      BET3I=TEMI
      IF(N3-100)14,18,18
   18 N4=N4+1
      Z(N4,1)=ALP4R
      Z(N4,2)=ALP4I
      N3=0
   41 IF(N4-N1)30,37,37
C  37 WRITE(6,555)
  555 FORMAT (//,2(11X,9HREAL PART,9X,9HIMAG PART,2X))
   37 CALL RTCK (ZRO,N8,Z)
C     WRITE(6,666) (Z(I,1),Z(I,2),I=1,N1)
  666 FORMAT (/,2(1PE22.7,1PE18.7))
      GO TO 300
30    IF(ABS (Z(N4,2))-1.E-5)10,10,31
   31 GO TO(32,10),L
   32 AXR=ALP1R
      AXI=-ALP1I
      ALP1I=-ALP1I
      M=5
      GO TO 99
   33 BET1R=TEMR
      BET1I=TEMI
      AXR=ALP2R
      AXI=-ALP2I
      ALP2I=-ALP2I
      M=6
      GO TO 99
   34 BET2R=TEMR
      BET2I=TEMI
      AXR=ALP3R
      AXI=-ALP3I
      ALP3I=-ALP3I
      L=2
      M=3
   99 TEMR=CCE(1)
      TEMI=0.0
      DO 100 I=1,N1
      TE1=TEMR*AXR-TEMI*AXI
      TEMI=TEMI*AXR+TEMR*AXI
```

```
  100 TEMR=      TE1+COE(I+1)
      HELL=TEMR
      BELL=TEMI
   42 IF(N4)102,103,102
  102 DO101I=1,N4
      TEM1=AXR-Z(I,1)
      TEM2=AXI-Z(I,2)
      TE1=TEM1*TEM1+TEM2*TEM2
      TE1A = TE1
      IF (TE1A .EQ. 0.0)  TE1A = 1.0E-35
      TE2=(TEMR*TEM1+TEMI*TEM2)/TE1A
      TEMI=(TEMI*TEM1-TEMR*TEM2)/TE1A
  101 TEMR=TE2
  103 GO TO(11,12,13,15,33,34),M
300   RETURN
      END


      SUBROUTINE RTCK (ZRO,N8,Z)
      REAL*8 XCOF(37),COF(37),ROOTR(36),ROOTI(36),DZR(36),DZI(36),
     1   COF1(37),COF2(37)
      DIMENSION ZRO(1),Z(60,2)
      M=N8-1
      DO 10 I=1,N8
10    XCOF(I) = ZRO(I)
      CALL DPOLRT (XCOF,COF,M,ROOTR,ROOTI,IER)
      IF (IER .EQ. 0)  GO TO 20
      WRITE (6,200)
200   FORMAT (//,' DPOLRT UNABLE TO FACTOR.  NO ROOT CHECK.',//)
      RETURN
C
C     FORM TWO NEW POLYNOMIALS FROM THE TWO SETS OF ROOTS
C
20    DO 21  I=1,M
      DZR(I) = Z(I,1)
21    DZI(I) = Z(I,2)
      CALL MAKPOL (M,DZR,DZI,COF1,COF)
      CALL MAKPOL (M,ROOTR,ROOTI,COF2,COF)
C
C     CALCULATE THE ERROR CRITERIA
C     THE SET OF ROOTS TO BE PRINTED OUT AND USED WILL BE THE  SET WHICH
C          WHEN EXPANDED YIELD THE MOST NEARLY CORRECT COEFFICIENTS WHEN
C          COMPARED (ABS VALUES) TO THE ORIGINAL POLYNOMIAL COEFFICIENTS
C
      ERR1 = 0.0
      ERR2 = 0.0
      DO 30 I=1,N8
      ERR1 =DABS(COF1(I)-XCOF(I)) + ERR1
```

```
30      ERR2 =DABS(CCF2(I)-XCCF(I)) + ERR2
        IF (ERR2 .GT. ERR1)  GO TO 50
        DO 40 I=1,M
        Z(I,1) = RCOTR(I)
40      Z(I,2) = ROOTI(I)
50      RETURN
        END
C


        SUBROUTINE MAKPCL (N,ROOTR,RCOTI,CR,CI)
        REAL*8  ROOTR(1),ROOTI(1),CR(1),CI(1)
C
C       TWO REAL*8 ARRAYS OF SIZE (N+1) MUST BE FURNISHED
C          FOR THE COEFFICIENTS CR (REAL PART) AND CI (IMAG PART)
C
C       N     NUMBER OF ROOTS
C       ROOTR   ARRAY OF REAL PARTS OF ROOTS
C       ROOTI   ARRAY OF IMAG PARTS OF ROOTS
C       CR      ARRAY OF REAL PARTS OF COEFF
C       CI      ARRAY OF IMAG PARTS OF COEFF
C       CR(N+1) = 1.0
C       CI(N+1) = 0.0
        IF ( N.LE. 0)  RETURN
        DO 10 I=1,N
        CR(I) = ROOTR(I)
10      CI(I) = ROOTI(I)
        CR(N+1) = 1.0
        CI(N+1) = 0.0
        K=N
        M=N-1
        DO 20  L=1,M
        DO 30  I=2,K
        CR(I) = CR(I) + CR(I-1)
30      CI(I) = CI(I) + CI(I-1)
        K= K-1
        DO 20  I=1,K
        J = I+L
        IF (DABS(ROOTI(J)).LE.1.D-35) RCOTI(J)=0.
        IF (DABS(ROOTR(J)).LE.1.D-35) ROOTR(J)=0.
        IF (DABS(CR(I)).LE.1.D-35) CR(I)=0.
        IF (DABS(CI(I)).LE.1.D-35) CI(I)=0.
        CR(I) = ROOTR(J)*CR(I) - ROOTI(J)*CI(I)
20      CI(I) = ROOTR(J)*CI(I) + ROOTI(J)*CR(I)
        K=N/2
        K=2*K/(N-K)
        DO 40  I=K,N,2
        CR(I) = -CR(I)
```

```
40      CI(I) = -CI(I)
        RETURN
        END
C


        SUBROUTINE OPT(NML,NEL,KEY1)
        DIMENSION NML(50,5)
        GO TO (4,5,5,6),KEY1
4       NS=1
        GO TO 7
5       NS=2
        GO  TO 7
6       NS=3
7       K1=1
        K2=2
        GO TO 3
1       K1=4
        K2=5
3       DO 109 J=NS,NEL
        IF(J-NS) 108,109,108
108     KK=J-1
        DO 121 K=NS,KK
        IF(NML(J,K1)-NML(K,K1))1121,103,1121
103     IF(NML(J,K2)-NML(K,K2))121,104,121
1121    IF(NML(J,K1)-NML(K,K2))121,122,121
122     IF(NML(J,K2)-NML(K,K1))121,104,121
104     DO 100 JJ=1,5
        N1=NML(J,JJ)
        LL=J-K-1
        IF(LL)100,100,107
107     DO 101 L=1,LL
        L1=J-L+1
101     NML(L1,JJ)=NML(L1-1,JJ)
100     NML(K+1,JJ)=N1
        GO TO 109
121     CONTINUE
109     CONTINUE
        IF(K1-1)2,1,2
2       RETURN
        END
C


        SUBROUTINE PARTS (NA,A,MK,AE,NK,AO,M1K,AE1,N1K,AO1)
        DIMENSION A(60),AE(30),AO(30),AE1(30),AO1(30)
        I=1
        MK=1
```

```
      NK=0
      M1K=0
      AE(1)=A(1)
      IF(NA-I)3,3,1
 1    I=I+1
      NK=NK+1
      AO(NK)=A(I)
      DUMMY=I-1
      AO1(NK)=DUMMY*A(I)
      IF(NA-I)3,3,2
 2    M1K=MK
      MK=MK+1
      I=I+1
      AE(MK)=A(I)
      DUMMY=I-1
      AE1(M1K)=DUMMY*A(I)
      IF(NA-I)3,3,1
 3    N1K=NK
      RETURN
      END
C

      SUBROUTINE SORT(ML,NN,NM,NE,NEL,NAL,NALL,KEY1,NML,
     1JI,KI,JO,KO)
      DIMENSION ML(50,5),NML(50,5)
      NALL=NAL
      GO TO(20,21,21,22),KEY1
   20 NM=1
      NEL=NE
      GO TO 23
   21 NM=2
      NEL=NE+1
      GO TO 23
   22 NM=3
      NEL=NE+2
   23 DO 100 J=NM,NEL
      JJ=J+1-NM
      DO 101 K=1,5
  101 ML(J,K)=NML(JJ,K)
  100 CONTINUE
      GO TO(1,2,2,2),KEY1
    1 RETURN
C     REMOVE ELEMENTS IN PARALLEL WITH INPUT
    2 DO 109 J=NM,NEL
      IF(ML(J,1)-JI)110,111,110
  111 IF(ML(J,2)-KI)109,112,109
  110 IF(ML(J,1)-KI)109,113,109
```

```
  113 IF(ML(J,2)-JI)109,112,109
C     INSERT LAST ELEMENT IN PLACE OF REMOVED ELEMENT
  112 DO 102 K=1,5
  102 ML(J,K)=ML(NEL,K)
      NEL=NEL-1
      GO TO 2
  109 CONTINUE
      GO TO(1,3,4,4),KEY1
C     INSERT TEST ELEMENT ACROSS INPUT
    3 ML(1,1)=JI
      ML(1,2)=KI
      ML(1,3)=99
      ML(1,4)=JI
      ML(1,5)=KI
      RETURN
C     REMOVE ELEMENTS IN PARALLEL WITH OUTPUT
    4 DO 115 J=NM,NEL
      IF(ML(J,4)-JO)117,116,117
  116 IF(ML(J,5)-KO)115,119,115
  117 IF(ML(J,4)-KO)115,120,115
  120 IF(ML(J,5)-JO)115,119,115
C     INSERT LAST ELEMENT IN PLACE OF REMOVED ELEMENT
  119 DO 105 K=1,5
  105 ML(J,K)=ML(NEL,K)
      NEL=NEL-1
      GO TO 4
  115 CONTINUE
      GO TO(1,3,5,6),KEY1
    5 ML(1,1)=JI
C     IN VOLTAGE GRAPH
      ML(1,2)=KI
      ML(1,3)=98
      ML(1,4)=JO
      ML(1,5)=KO
      NALL=NALL+1
      RETURN
C     INSERT TEST ELEMENT ACROSS INPUT AND OUTPUT
    6 ML(2,1)=JI
      ML(2,2)=KI
      ML(2,3)=97
      ML(2,4)=JI
      ML(2,5)=KI
      ML(1,1)=JO
      ML(1,2)=KO
      ML(1,3)=96
      ML(1,4)=JO
      ML(1,5)=KO
      RETURN
```

```
      END
C
      FUNCTION  SUM (N,A,W)
      DIMENSION A(30)
      SUM =0.0
      IF(N)3,3,2
 2    SUM =A(N)
      IF(N-1)3,3,4
 4    X=-W*W
      NM1=N-1
      DO 5 I=1,NM1
      K=N-I
 5    SUM =SUM *X+A(K)
 3    RETURN
      END
C
      SUBROUTINE TEST(MG,ML,NEL,NALL,LN,KYOUT,K10)
      DIMENSION MG(30,5),ML(50,5),MM(50)
C     THIS TEST IS MADE BY TRYING TO ADD ONE NEW NODE AT A TIME UNTIL
C     ALL NODES HAVE BEEN ACCOUNTED FOR (SEE TRETST)
2     K1=1
      K2=2
      GO TO 11
25    K1=4
      K2=5
11    DO 21 J=1,NEL
21    MM(J)=C
      K11=LN
32    DO 30 J=K10,NEL
      K11=K11+1
      MG(K11,3)=ML(J,3)
      MG(K11,K1)=ML(J,K1)
30    MG(K11,K2)=ML(J,K2)
      K=1
      M=MG(1,K1)
      N=MG(1,K2)
      MM(M)=1
      MM(N)=1
9     KK=K
      DO 7 J=2,K11
      M=MG(J,K1)
      N=MG(J,K2)
      IF(MM(N))1,1,3
1     IF(MM(M))7,7,4
```

```
3       IF(MM(M))4,4,7
4       MM(M)=MM(M)+1
        MM(N)=MM(N)+1
        K=K+1
        IF(K-LN)7,10,10
7       CONTINUE
        IF(KK-K)9,8,9
8       KYOUT=-1
        RETURN
10      IF(NALL)17,17,110
110     IF(K1-4)25,17,25
17      KYOUT=1
        RETURN
        END
C


        SUBROUTINE TOPOL(NPL,NAL,NN,JI,KI,JO,KO,KEY1,KEY2,MP,ELT,VAL,
       1MAP,ELTA,VALA,Y1,Y2,VALL,KW,NVAR,KEY3,Y,NP,JP)
        DIMENSION VAL (100)
        DIMENSION MP(100,3),ML(50,5),ELT(100),MAP(20,5),ELTA(20),VALA(20),
       1C(50),G(50),H(50),MG(30,5),NG(50),NML(50,5),              X(60),Y(
       260),Y1(60),Y2(60),VALL(50)
        REAL IH1 /4H-    /, IH2 / 4H+    /
C       IF NETWORK FUNCTION HAS ALREADY BEEN CALCULATED IN BILINEAR FORM,
C       CHANGE ONLY PART OF FUNCTION
        GO TO (6,102),KEY3
102     IF(NV)131,131,130
131     NVV=MP(NVAR,3)
        GO TO 132
130     NVV=MAP(NV,5)
        GO TO 133
132     IF(C(NVV))133,134,133
133     DO 135 J=1,NP
135     Y1(J)=VALL(KW)*Y1(J)/VALL(KW-1)
        GO TO 48
134     DO 136 J=1,NP
136     Y1(J)=VALL(KW-1)*Y1(J)/VALL(KW)
        GO TO 48
C       COMBINE PARALLEL RLC ELEMENTS INTO SINGLE TOPOLOGICAL ELEMENT
6       CALL GROUP(G,C,H,NML,MP,MAP,ELT,ELTA,VAL,VALA,NPL,NAL,NE)
C       INSERT TEST ELEMENTS ACROSS INPUT,OUTPUT
        CALL SORT(ML,NN,NM,NE,NEL,NAL,NALL,KEY1,NML,JI,KI,JO,KO)
        CALL OPT(ML,NEL,KEY1)
        GO TO (103,103,103,103,104),KEY2
C       PRINT ELEMENT NUMBER,CORRESPONDING TOPOLOGICAL ELEMENT NUMBER
  104   WRITE (6,80)
        WRITE (6,105) (J,MP(J,3),J=1,NPL)
```

```
80    FORMAT(//,1X,19HELEMENT ASSOCIATION  ,/,1X,8HPHYSICAL  ,1X,11HTOPO
     1LOGICAL ,//)
105   FORMAT(I5,8X,1HY,I2)
      IF(NAL)106,81,106
  106 WRITE (6,105) (J,MAP(J,5),J=1,NAL)
   81 WRITE (6,82)
82    FORMAT(//,1X,25HTREES,2-TREES,OR 3-TREES  ,//)
103   LN=NN-1
      NV=NVAR-NPL
      KK=0
      NP=2*LN+1
      DO 274 J=1,NP
      Y1(J)=0.
      Y2(J)=0.
274   Y(J)=0.
      LM=NN-2
      MY=0
      MK=NEL+2-NN
      L=NN-1
      I=1
C     PICK FIRST SET OF ELEMENTS TO BE TESTED AS TREE
      DO 5 K=1,LN
      MAX=MAX+K
      NG(K)=K
      DO 8 J=1,5
8     MG(K,J)=ML(K,J)
5     CONTINUE
      KY1N=0
C     TEST FOR TREE IN CURRENT GRAPH
100   CALL TRETST(NN,LN,KY1N,KYOUT,LL,MG)
      IF(KYOUT)17,17,18
   17 IF(MY)21,20,21
   21 MY=0
      KY1N=0
      GO TO 121
   20 KY1N=-1
C     IF CIRCUIT HAS BEEN FORMED,REMOVE FATAL ELEMENT FROM LIST
121   IF(LL)272,19,272
272   IF(LL-LN)281,19,19
281   IF(KK-1-NEL)282,19,19
282   L=LL+1
      GO TO 50
C     IF NETWORK IS ACTIVE,TEST VOLTAGE GRAPH
18    IF(NALL)233,235,233
235   II=1
      GO TO 899
  233 IF(MY)35,34,35
   34 MY=1
```

```
      KY1N=1
      GO TO 10C
C     IF NETWORK IS ACTIVE,TEST FOR SIGN OF TREE
   35 MY=0
      KY1N=0
      II=IGN(NN,LN,MG)
899   I=I+1
857   IF(II)870,871,871
870   AA=IH1
      GO TO 872
871   AA=IH2
C     ASSEMBLE TREE-ADMITTANCE PRODUCT
872   CALL ASBS(LN,C,G,H,X,NP,MG,I,KEY1,JN,JP,II,NPL,NAL)
C     IF TO BE CALCULATED IN BILINEAR FORM,TEST FOR VARIABLE ELEMENT
      IF(NVAR)146,45,146
146   IF(NV)347,347, 46
347   DO 147 K=JP,LN
      IF(MG(K,3)-MP(NVAR,3))147,44,147
147   CONTINUE
      GO TO 45
46    DO 148 K=JP,LN
      IF(MG(K,3)-MAP(NV,5))148,44,148
148   CONTINUE
      GO TO 45
44    DO 144 J=1,NP
144   Y1(J)=Y1(J)+X(J)
      GO TO 149
45    DO 145 J=1,NP
145   Y2(J)=Y2(J)+X(J)
149   GO TO (19,19,19,19,89),KEY2
C     PRINT SYMBOLIC TREE ADMITTANCE PRODUCT
   89 WRITE (6,900) AA, (MG(J,3),J=JP,LN)
900   FORMAT(1X,A1,2X,20(1HY,I2,2X)/(21(1HY,I2,2X)))
C     PERMUTE ELEMENT
19    IF(NG(1)-MK)47,48,48
47    IF(KK-1-NEL)747,50,50
747   K=NG(1)
      DO 265 NY=1,5
  265 MG(1,NY)=ML(K,NY)
      IF(NG(L)-NEL)49,50,50
49    NG(L)=NG(L)+1
      K=NG(L)
      DO 77 J=1,5
77    MG(L,J)=ML(K,J)
      GO TO 100
C     PERMUTE NEXT ELEMENT IN LIST
C
C  EXAMINE THE ORIGINAL IN THIS AREA
```

```
C
   50 L = L-1
      NG(L) = NG(L) + 1
      DO 51 K=L,LM
   51 NG(K+1)=NG(K)+1
      KK=NG(L)
      DO 52 J=L,LN
      DO 267 NY=1,5
  267 MG(J,NY)=ML(KK,NY)
52    KK=KK+1
C     IF 2-TREE OR 3-TREE,SEE IF TEST ELEMENTS HAVE BEEN PERMUTED
      GO TO (427,428,428,429),KEY1
428    IF(NG(1)-1)427,427,48
429   IF(NG(2)-2)427,427,48
427   IF(KK-1-NEL)53,100,100
C     AFTER PERMUTATION,TEST FOR CONNECTED GRAPH
53    CALL TEST(MG,ML,NEL,NALL,LN,KYOUT,KK)
      IF(KYOUT)1050,953,953
1050  IF(L-1)48,48,50
953   L=NN-1
      GO TO 100
C     END TREE TEST,ASSEMBLE AND SHIFT COEFFICIENTS
48    DO 550 J=1,NP
550   Y(J)=Y1(J)+Y2(J)
      IF(JP-1)331,331,555
555   JW=JP-1
      NP=NP+2*JW
      DO 556 J=1,NP
      K=NP-J+1
      KK=K+JW
556   Y(KK)=Y(K)
      DO 557 J=1,JW
      NW=NP-J+1
      Y(J)=0.
557   Y(NW)=0.
  331 RETURN
      END
C


      SUBROUTINE TRETST(NN,LN,KY1N,KYCUT,LL,MG)
      DIMENSION MG(30,5),MM(50) ,MMM(50)
C     THIS TEST IS MADE BY TRYING TO ADD ONE NEW NODE AT A TIME UNTIL
C     ALL NODES HAVE BEEN ACCOUNTED FOR
      LL=0
      IF(KY1N)11,2,25
2     K1=1
      K2=2
```

```
      GO TO 11
25    K1=4
      K2=5
   11 DO 21 J=1,NN
      MMM(J)=0
   21 MM(J)=C
      K=1
      M=MG(1,K1)
      N=MG(1,K2)
      MM(M)=1
      MM(N)=1
    9 KK=K
      DO 7 J=2,LN
      M=MG(J,K1)
      N=MG(J,K2)
C     IF ONE NEW NODE IS ADDED,STORE ELEMENT NUMBER
C     IF NO NEW NODE IS ADDED AND ON FIRST PASS,CIRCUIT IS PRESENT,
C     STORE NUMBER OF FATAL CIRCUIT ELEMENT AND RETURN
C     IF TWO NEW NODES ARE ADDED,GO TO NEXT ELEMENT
      IF(MM(N))1,1,3
    1 IF(MM(M))7,7,4
    3 IF(MM(M))4,4,13
13    IF(KK-1)7,14,15
15    IF(MG(J,3)-MMM(J))8,7,8
14    LL=J
      GO TO 8
    4 MM(M)=MM(M)+1
      MM(N)=MM(N)+1
      MMM(J)=MG(J,3)
      K=K+1
      IF(K-LN)7,10,10
   10 KYOUT=1
      RETURN
    7 CONTINUE
      IF(KK-K)9,8,9
    8 KYOUT=-1
      RETURN
      END
C


      SUBROUTINE PLOT(X,Y,NN)
C
C***  GUAGE INPUT & FIND MAX & MIN FOR X & Y; CALL UTPLOT
C       X - THE X-AXIS COORDINATE
C       Y - THE Y-AXIS COORDINATE
C       NN- THE NUMBER OF POINTS TO BE PLOTTED
C
```

```
      DIMENSION X(NN), Y(NN), RANGE(4)
      EQUIVALENCE (RANGE(1),XMAX), (RANGE(2),XMIN),(RANGE(3),YMAX),
     1           (RANGE(4),YMIN)
      IF (NN .GE. 4)  GO TO 200
      WRITE (6,201)
201   FORMAT (10(/),10X,'GRAPHS OF LESS THAN FOUR POINTS ARE NOT PLOTTED
     1.')
      RETURN
200   WRITE (6,300)
  300 FORMAT (1H1)
      XMAX=-1.E20
      XMIN=1.E20
      YMAX=-1.E20
      YMIN=1.E20
      DO 1 I=1,NN
      IF(X(I)-XMAX) 6,6,2
 2    XMAX=X(I)
      YXMAX=Y(I)
 6    IF(X(I)-XMIN) 3,3,7
 3    XMIN=X(I)
      YXMIN=Y(I)
 7    IF(Y(I)-YMAX) 8,8,4
 4    YMAX=Y(I)
      XYMAX=X(I)
 8    IF(Y(I)-YMIN) 5,5,1
 5    YMIN=Y(I)
      XYMIN=X(I)
 1    CONTINUE
      CALL UTPLOT(X,Y,NN,RANGE,1)
      WRITE (6,101) YMAX, XYMAX,YMIN,XYMIN
101   FORMAT (/,10X,
     1          'MAX Y='1PE12.4,' AT X='E12.4,20X,'MIN Y=' E12.4,' AT X='
     2   E12.4)
      WRITE (6,100) XMAX, YXMAX, XMIN, YXMIN
100   FORMAT (/,10X,
     1          'MAX X='1PE12.4,' AT Y='E12.4,20X,'MIN X=' E12.4,' AT Y='
     2   E12.4)
      RETURN
      END
C                              SECTION (3)
C
C     THIS SECTION PRODUCES THE COST FUNCTION (FUNC) AND THE DISCRETE
C     APPROXIMATION TO THE GRADIENT VECTOR (DER(II))
C

      SUBROUTINE GRAD(NPL,NAL,NN,JI,KI,JO,KO,MP,ELT,VAL,LIN,NOM,
     1OMGMIN,NVAR,CMGMAX,D,DER,FUNC,KEY1,KY,WM)
```

```
      DIMENSION MP(100,3),ML(50,5),ELT(100),MAP(20,5),ELTA(20),
     1VAL(100),VALA(20),C(50),G(50),H(50),Y11(60),Y12(60),Y(60),Z(60)
     2,Y21(60),Y22(60),VALL(50) ,ZZ(60,2),D(100),R2(100),PP(60,2)
     3,DR2(100),TVAL(100),DER(100),TP(100),WM(100)
      JW=1
      II=-1
      ND=NOM+1
      DEL=(OMGMAX-OMGMIN)/NOM
      GO TO 6
  100 TVAL(II)=VAL(II)
      VAL(II)=1.0001*VAL(II)
    6 CALL TOPOL(NPL,NAL,NN,JI,KI,JO,KO,3,2,MP,ELT,VAL,MAP,ELTA,
     1VALA,Y11,Y12,VALL,JW,0,1,Y,NP,J11)
   10 CALL TOPOL(NPL,NAL,NN,JI,KI,JO,KO,2,2,MP,ELT,VAL,MAP,ELTA,
     1VALA,Y21,Y22,VALL,JW,0,1,Z,NZ,J22)
114   DO 20 J=1,60
      IF(Y(1))16,15,16
15    IF(Z(1))16,17,16
16    JP=NP-J+1
      JZ=NZ-J+1
      GO TO 21
17    DO 18 K=1,NP
18    Y(K)=Y(K+1)
      DO 19 K=1,NZ
19    Z(K)=Z(K+1)
20    CONTINUE
C     CALCULATE DEGREE OF NUMERATOR,DENOMINATOR,IGNORING HIGHER DEGREE
C     ZERO COEFFICIENTS
   21 CONTINUE
      DO 30 J=1,60
      JJ=JP-J+1
      IF(Y(JJ))32,30,32
32    JP=JP-J+1
      GO TO 34
30    CONTINUE
34    DO 35 J=1,60
      JJ=JZ-J+1
      IF(Z(JJ))33,35,33
33    JZ=JZ-J+1
      GO TO 36
35    CONTINUE
C     CALCULATE ZEROS
   36 CALL MULLER(Y,JP,ZZ)
C     CALCULATE POLES
      CALL MULLER(Z,JZ,PP)
C     CALCULATE FREQUENCY RESPONSE
   83 IF(II)82,82,102
   82 CALL FREQQ(LIN,NOM,OMGMIN,OMGMAX,JP,JZ,Y,Z,KEY1,R2)
```

```
      FUNC=0.
C     CALCULATE THE COST FUNCTION
      DO 99 J=1,ND
   99 FUNC=WM(J)*DEL*(R2(J)-D(J))**2+FUNC
      II=1
      GO TO (104,105),KY
  105 GO TO(100,104),KEY1
  102 CALL FREQQ(LIN,NOM,OMGMIN,OMGMAX,JP,JZ,Y,Z,KEY1,DR2)
      DF=0.
C     CALCULATE THE COST FUNCTION FOR THE PERTURBATION OF THE VARIABLE
      DO 101 JJ=1,ND
  101 DF=WM(JJ)*DEL*(DR2(JJ)-D(JJ))**2+DF
C     CALCULATE THE PARTIAL DERIVATIVE FOR EACH VARIABLE
      DER(II)=(FUNC-DF)/(.0001*TVAL(II))
      VAL(II)=TVAL(II)
      II=II+1
      IF(II-NVAR)100,100,104
  104 RETURN
      END
```

LIST OF REFERENCES

1. Kuo, F. F., Network Analysis and Synthesis, John Wiley and Sons, Inc., 1962.

2. Wilde, D. J., Optimum Seeking Methods, Prentice-Hall, 1964.

3. Rosen, J. B., "The Gradient Projection Method for Nonlinear Programming. Part I. Linear Constraints," J. SIAM (1960).

4. Calahan, D. A., "Linear Network Analysis and Realization Digital Computer Programs: An Instruction Manual," University of Illinois Bulletin, Vol. 62, No. 58.

5. Weinberg, L., Network Analysis and Synthesis, McGraw-Hill, 1962.

6. Kirk, D. E., Optimal Control Theory: An Introduction, Prentice-Hall, to be published 1970.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	20
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Commandant of the Marine Corps (Code A03C) Headquarters, U. S. Marine Corps Washington, D. C. 20380	1
4.	James Carson Breckinridge Library Marine Corps Development and Educational Command Quantico, Virginia 22134	1
5.	Professor Donald E. Kirk Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	3
6.	MAJ Charles A. Henry 4807 Troy Lane La Mesa, California 92041	1
7.	Professor S. G. Chan Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
8.	Mr. H. Karl Bouvier Jet Propulsion Laboratory 4800 Oak Grove Drive Pasadena, California 91103	2

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Naval Postgraduate School Monterey, California 93940	Unclassified
	2b. GROUP

3 REPORT TITLE

Computer-Aided Design of Passive Filters in the Frequency Domain Using Gradient-Projection Minimization Techniques

4 DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Master's Thesis, December 1969

5. AUTHOR(S) (First name, middle initial, last name)

Charles Alden Henry

6 REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
December 1969	104	6

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Naval Postgraduate School Monterey, California 93940

13. ABSTRACT

An algorithm for the computer-aided design of passive electrical filters using minimization techniques is presented. The configuration, or topology of the filter and its desired frequency response are specified. A measure of the deviation between the desired and the actual frequency response is minimized using the gradient projection search method. Examples are presented which demonstrate the performance of the minimization procedure.

DD FORM 1 NOV 68 1473 (PAGE 1)

S/N 0101-807-6811

Security Classification

A-31408

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Passive Filter						
Computer-Aided						
Frequency Response Optimization						